Chapter 4

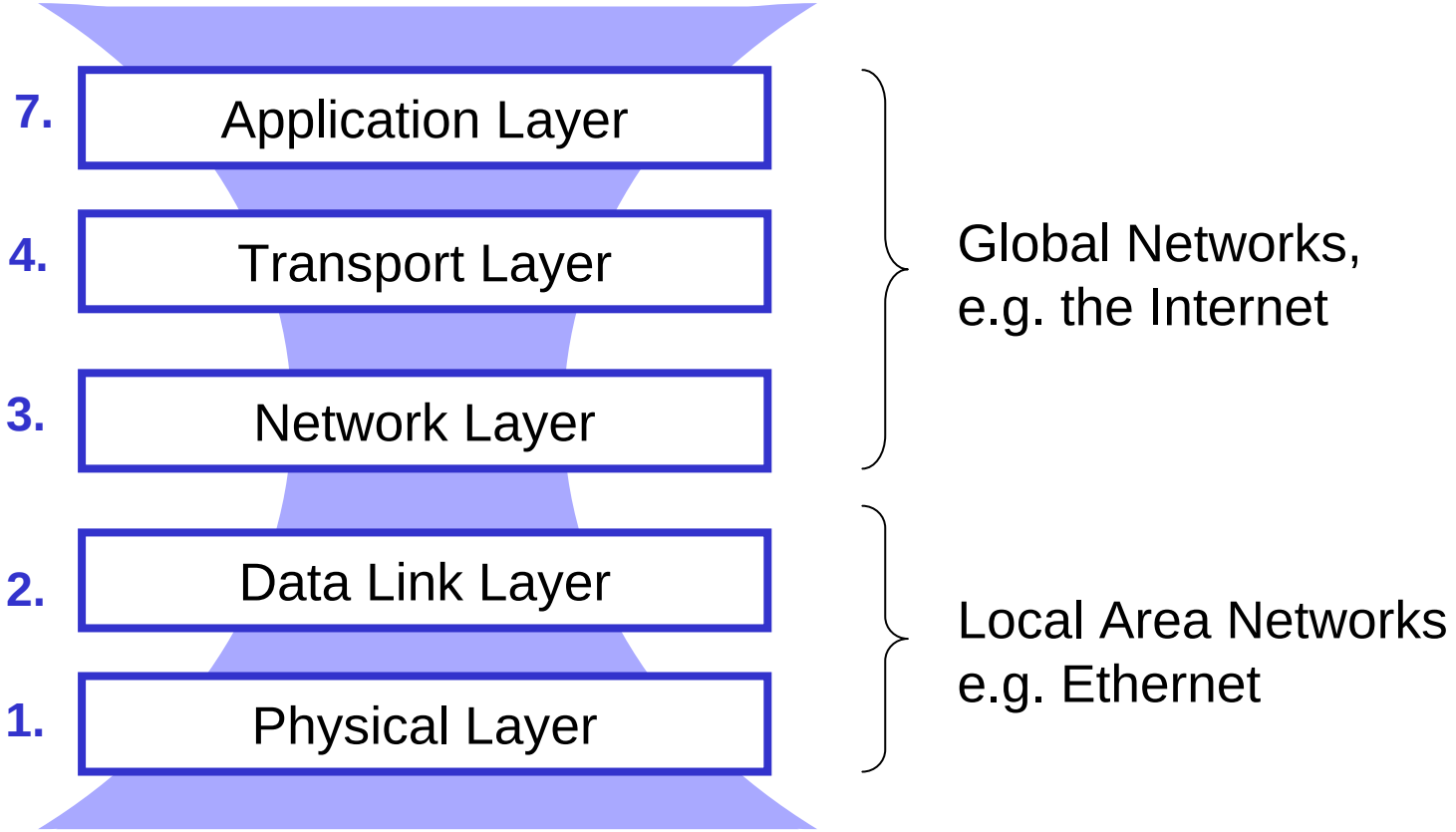
Advanced computer networking

Internet Protocols

Thomas Fuhrmann

Network Architectures
Computer Science Department
Technical University Munich

From Local to Global Networks



RFC 791 – IPv4

December 1981

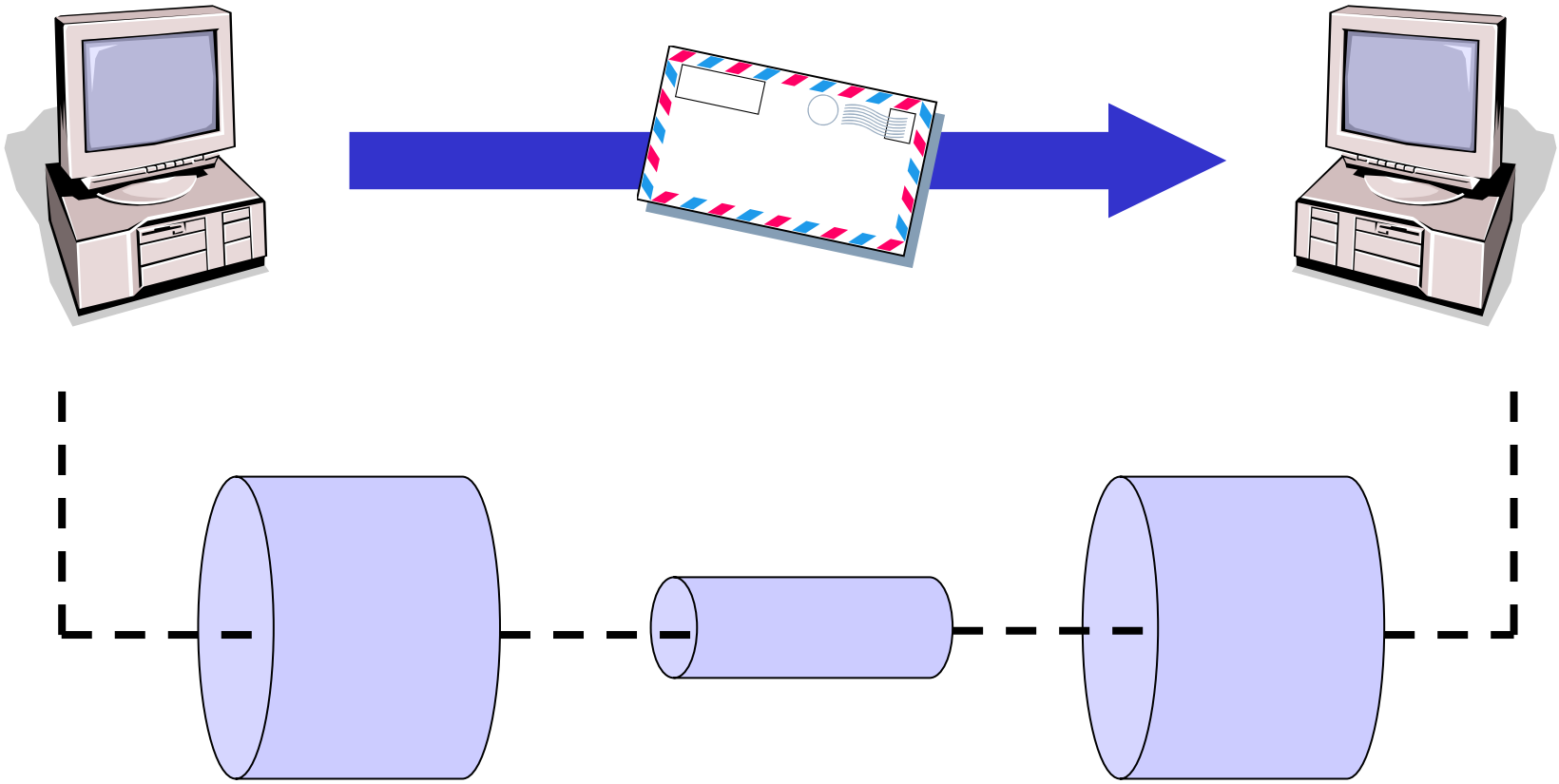
The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. [..] The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.

The internet protocol is specifically limited in scope to provide the functions necessary to deliver [..] an internet datagram from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols.

RFC 791 – IPv4 (Discussion)

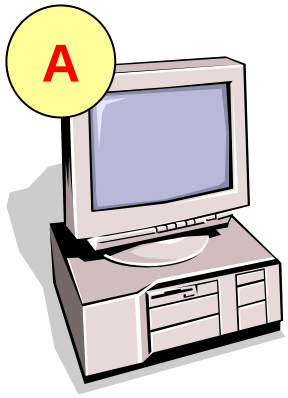
- The Internet Protocol provides
  - Host-to-host communication across interconnected systems
  - (Global) datagram service for packet switching networks
  - Fragmentation & reassembly
- It does not provide
  - End-to-end reliability
  - Flow control
  - Sequencing
- Note that IP is designed to sit between the data link layer and the transport layer.
  - Source and destinations with (globally unique) addresses.
  - Transmission by (local) datagram networks.
  - No services that can be provided by the terminals themselves.
- The Internet Protocol is the smallest common denominator for global connectivity across different networks.

Fragmentation & Reassembly

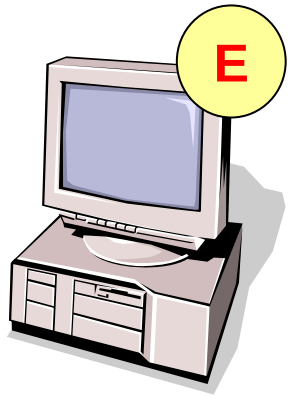


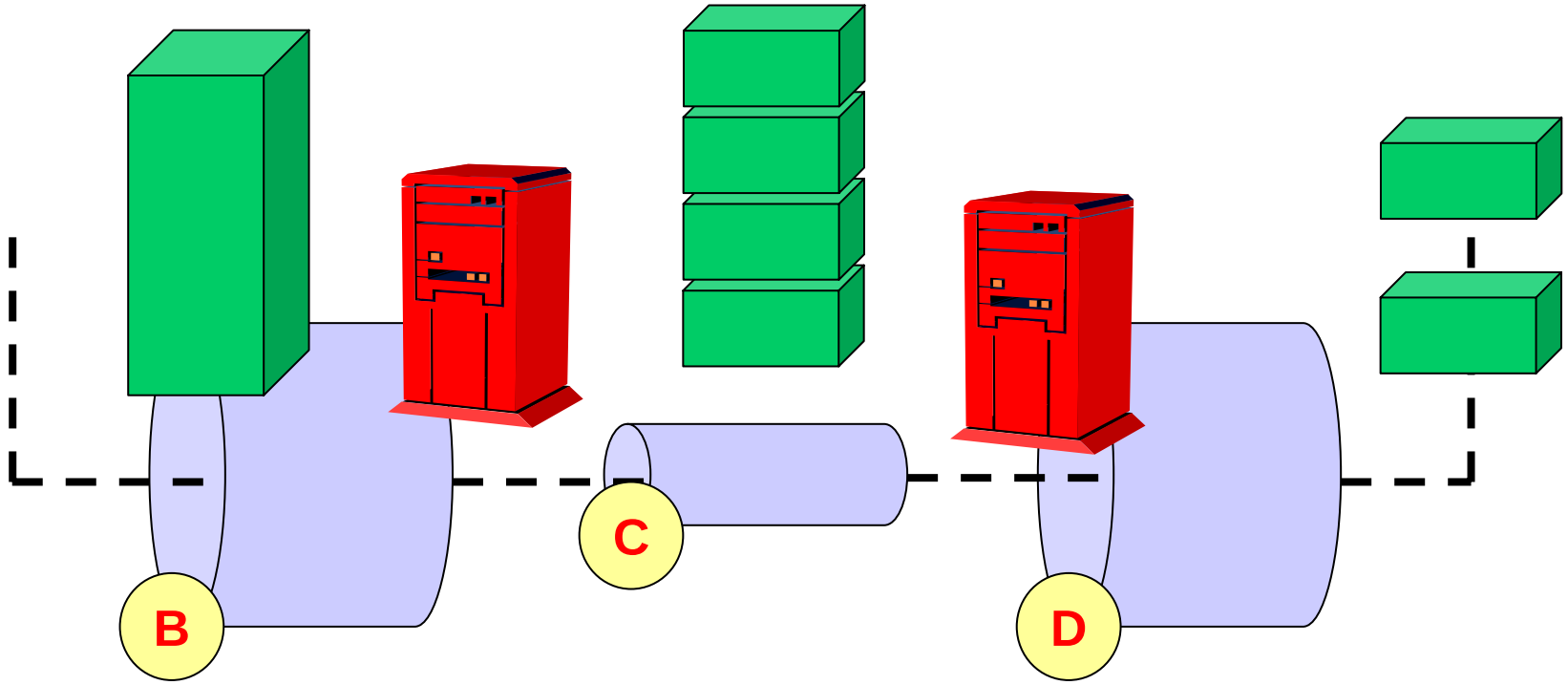
Data link layer offers
(unreliable) datagram service, but the datagram size may be limited. – Furthermore, different layer 2 technologies may have different maximum transport unit (=MTU).

Fragmentation & Reassembly

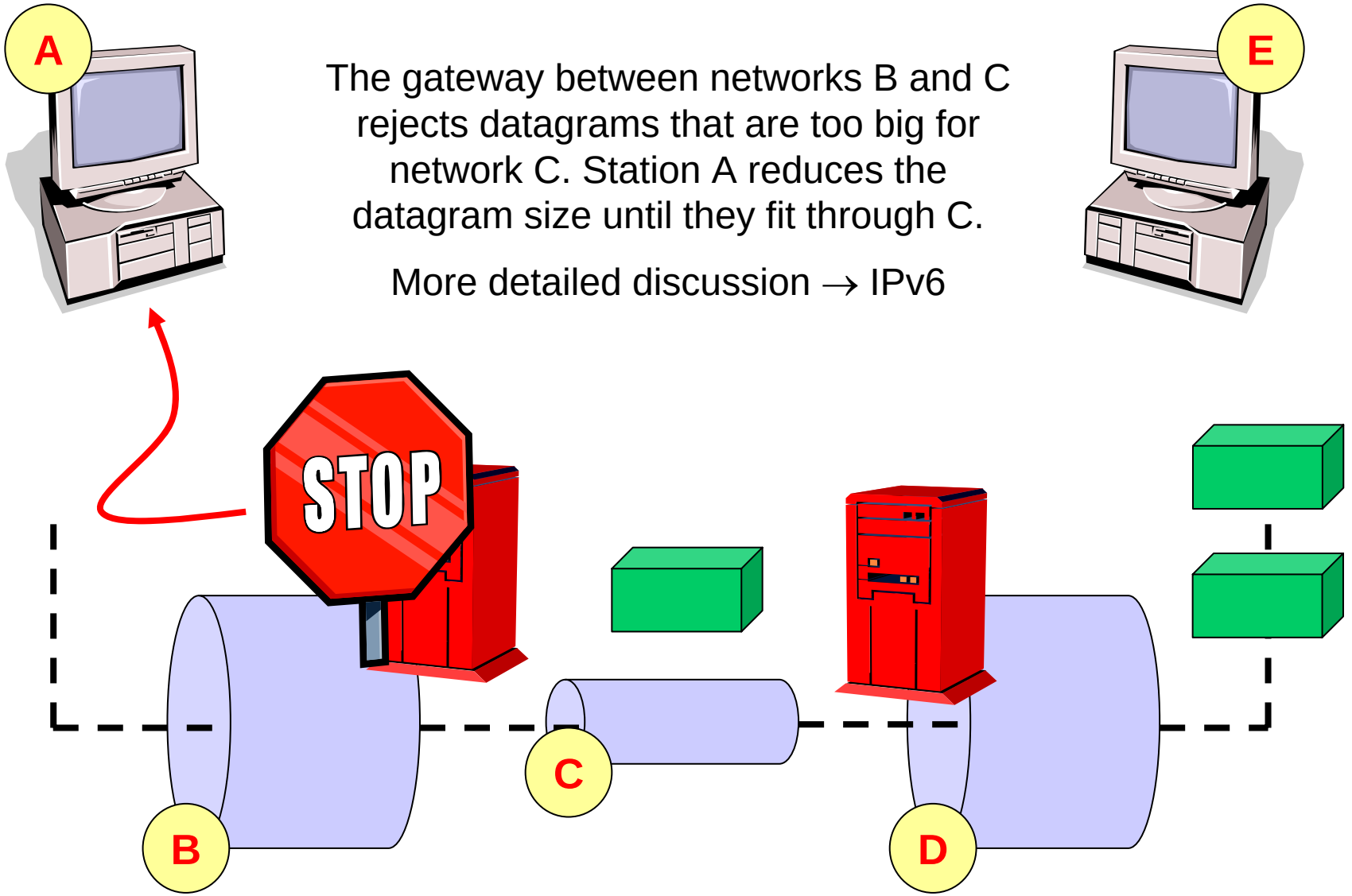


Station A send a datagram that is too big for network C. The gateway between networks B and C splits the datagram into several fragments. The station E reassembles these fragments.





An Alternative: Path MTU Discovery



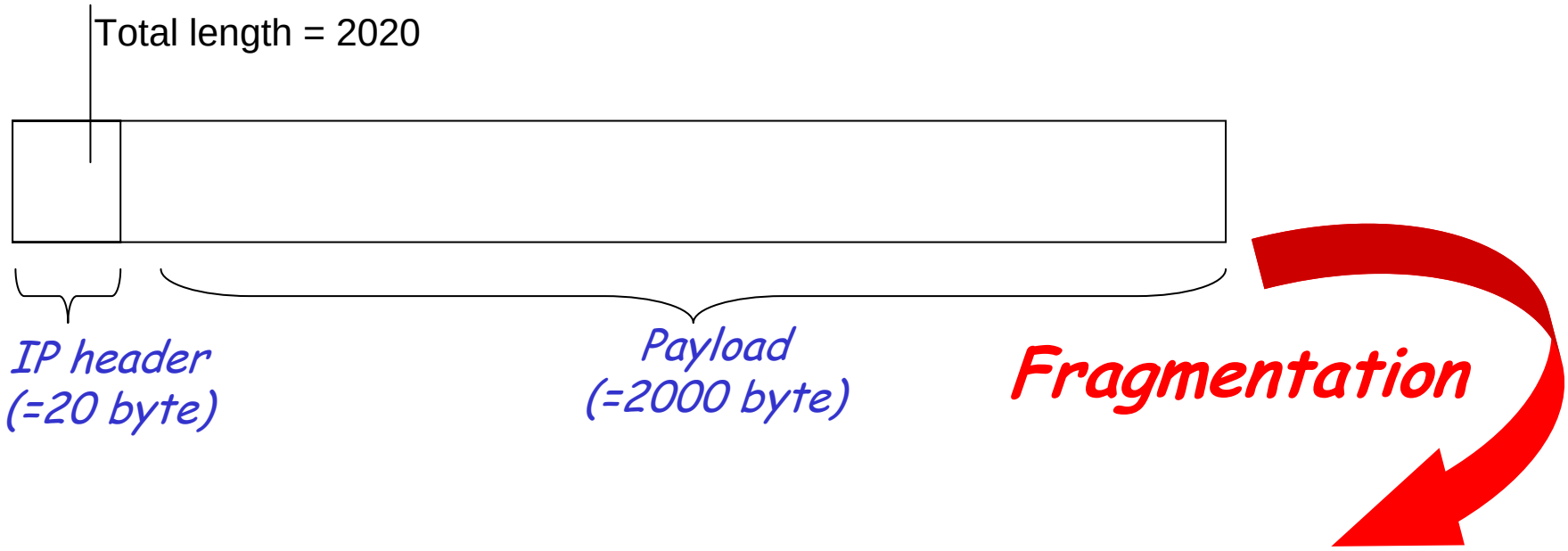
Internet Protocol Version 4

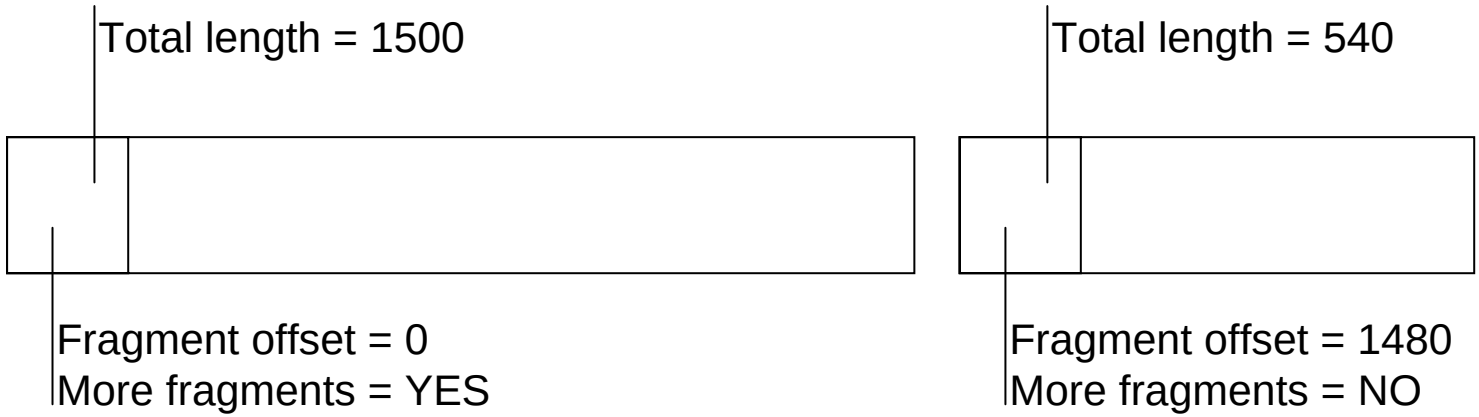
+	Bits 0 - 3	4 - 7	8 - 15	16 - 18	19 - 31
0	Version	Header length	Type of Service (now DiffServ and ECN)	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
160/192+	Data				

IPv4 Header Fields Explained

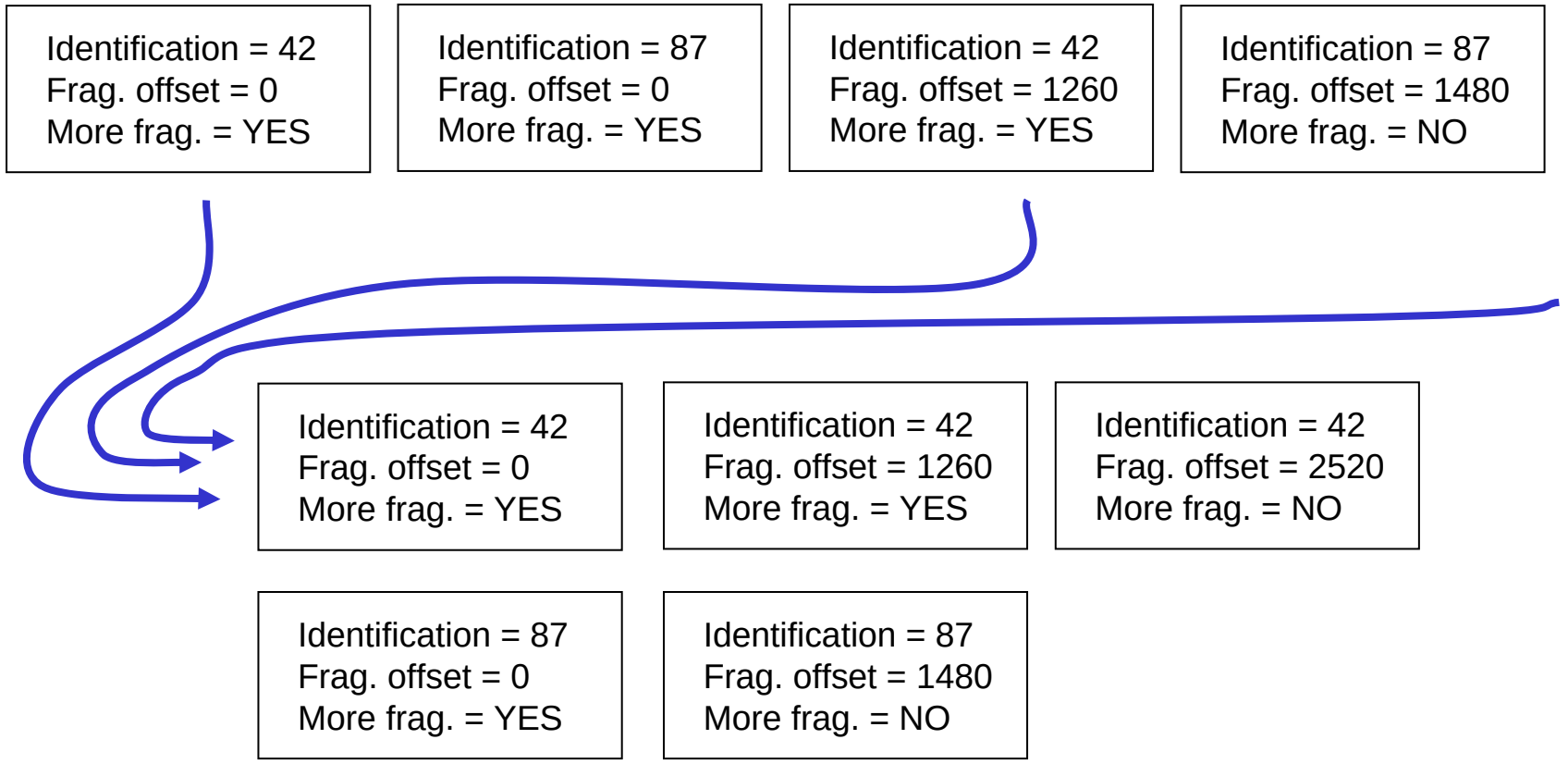
- IP Version - Differentiate protocol versions on the network layer.
- Header Length – Allow optional header fields.
- Type of service (=TOS)
  - 8 priority levels: Do not forward low priority packets when there are not enough resources.
  - Normal / low delay
  - Normal / high throughput
  - Normal / high reliability
  - Minimal monetary cost (cf. RFC 1349)
- Note that meanwhile, the TOS bits are obsolete → ‚differentiated services' RFC 2474 and ‚early congestion notification' RFC 3168.
- Total length – Allow variable packet size
  - Some network layer technologies (such as ATM) have fixed packet size.
  - The maximal IP packet size is 64 KByte
- Identification
  - Allows to distinguish the packets a host sends.
  - Need to reassemble the fragments of a packet.
- Flags
  - Do not fragment → Used for path MTU discovery
  - More fragments
- Fragment offset
  - Allows to reassemble the fragments in the right order
- Time to live (TTL)
  - Prevent infinite routing loops
  - Set at sending host, decrement in each router, discard packet when TTL reaches zero.
- Protocol – Differentiate different transport layer protocols
- Header checksum – The sum of all 16bit words in the header

Fragmentation Example





Fragmentation Example (cont.)



The 16 bit identification together with the source address must uniquely identify the original packet so that the fragments can be correctly reassembled.

1. Fragmenting IP Packets

2. Options in the IP header

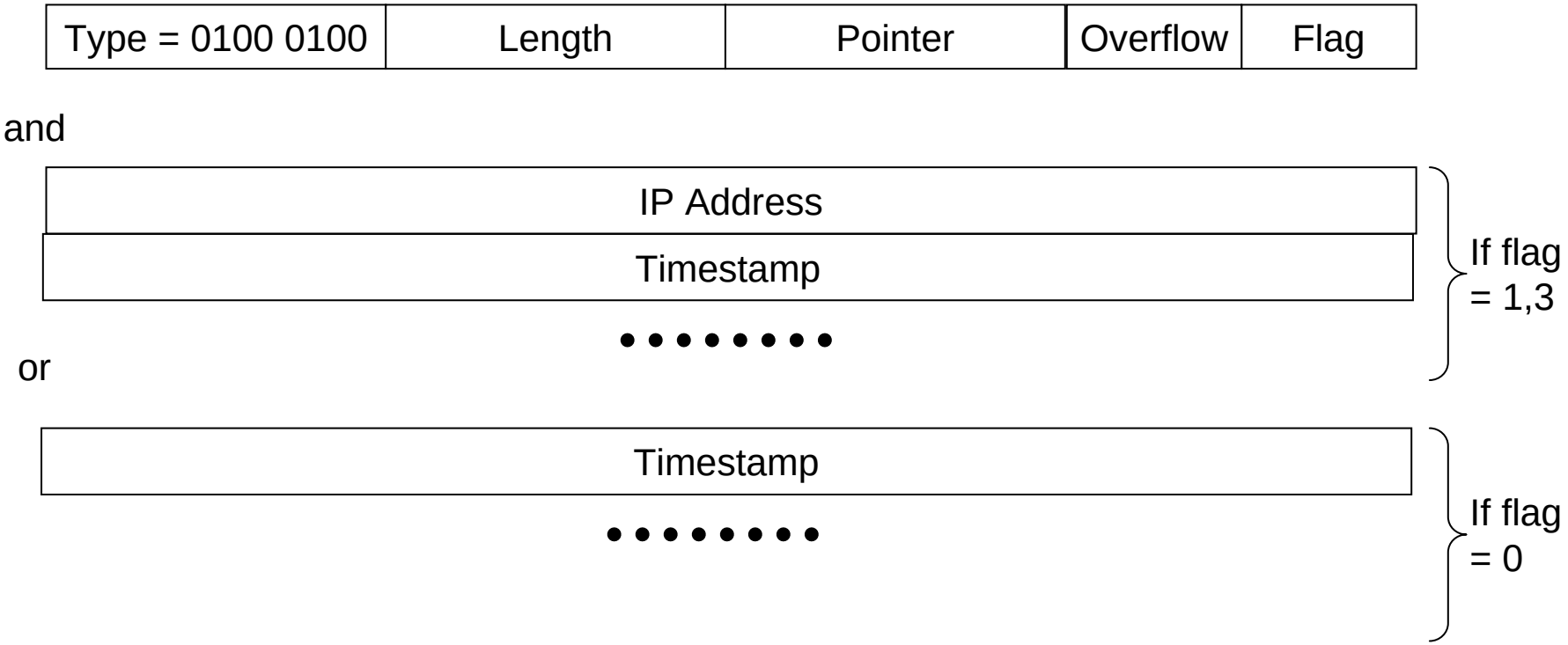
3. Global versus Local Addresses

4. Routing

IP Options – Overview

- IP header can be supplemented with options. RFC 791 specifies:
  - End of Option list (occupies only one octed, no length octet).
  - No Operation (occupies only one octed, no length octet).
  - Security. Used to carry Security, Compartmentation, User Group, and Handling Restriction Codes.
  - Loose Source Routing. Used to route the internet datagram based on information supplied by the source.
  - Strict Source Routing. Used to route the internet datagram based on information supplied by the source.
  - Record Route. Trace the route an internet datagram takes.
  - Stream ID (obsolete)
  - Internet Timestamp. Record timestamps along the datagram's route.
- Each IP header may contain several options.
- Options can have variable length.
- Insert an ‚end of option list' if end of options does not coincide with end of IP header.

Timestamp Option



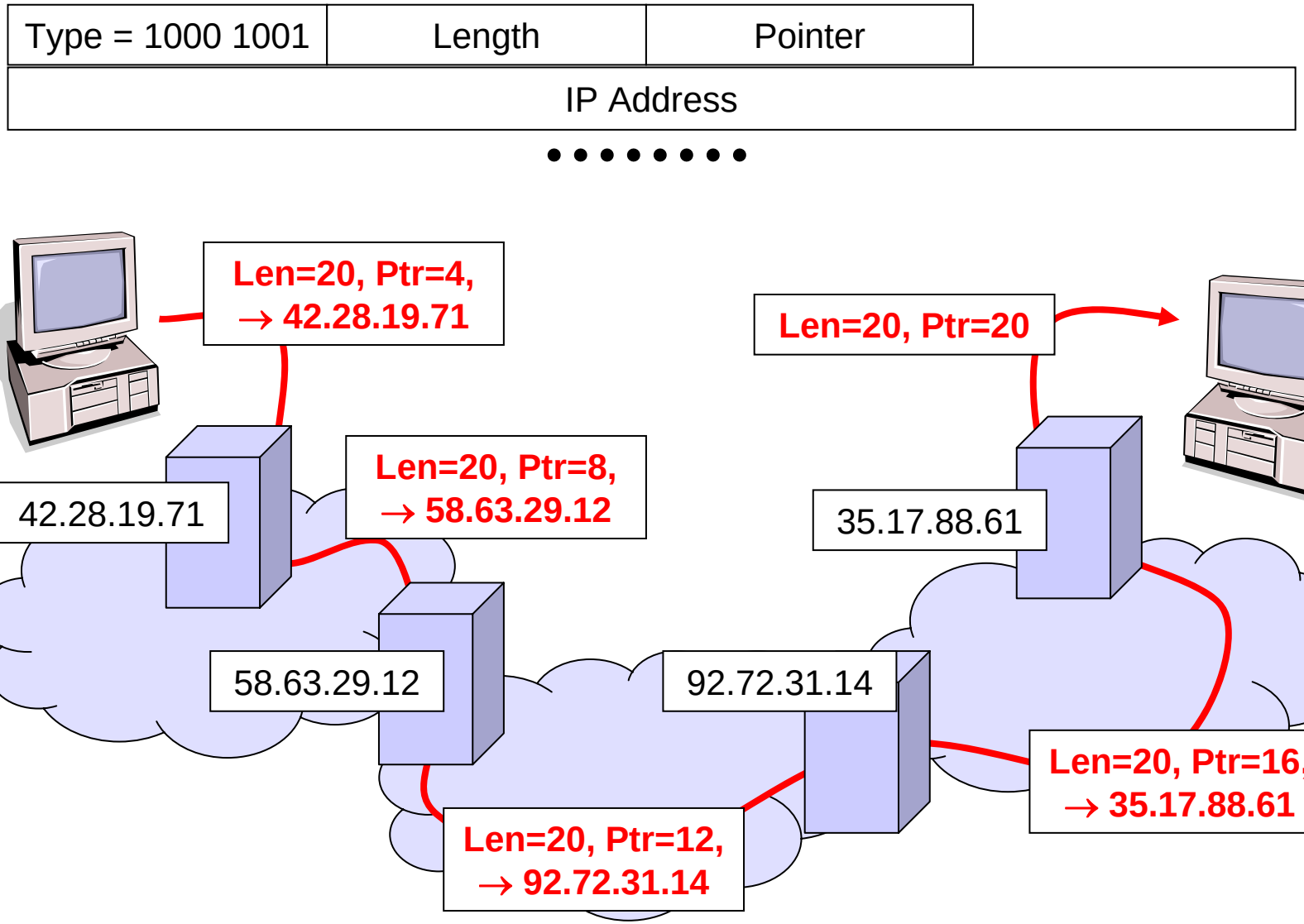
Record timestamps during forwarding. Pointer points to next entry to fill, i.e. list is full when pointer exceeds length. Then the overflow counter is increased.

Flag = 0, record timestamps only

Flag = 1, record IP address and timestamp

Flag = 3, preset IP address, record timestamp only when address is in list

Strict Source Route Option



Comments on Options

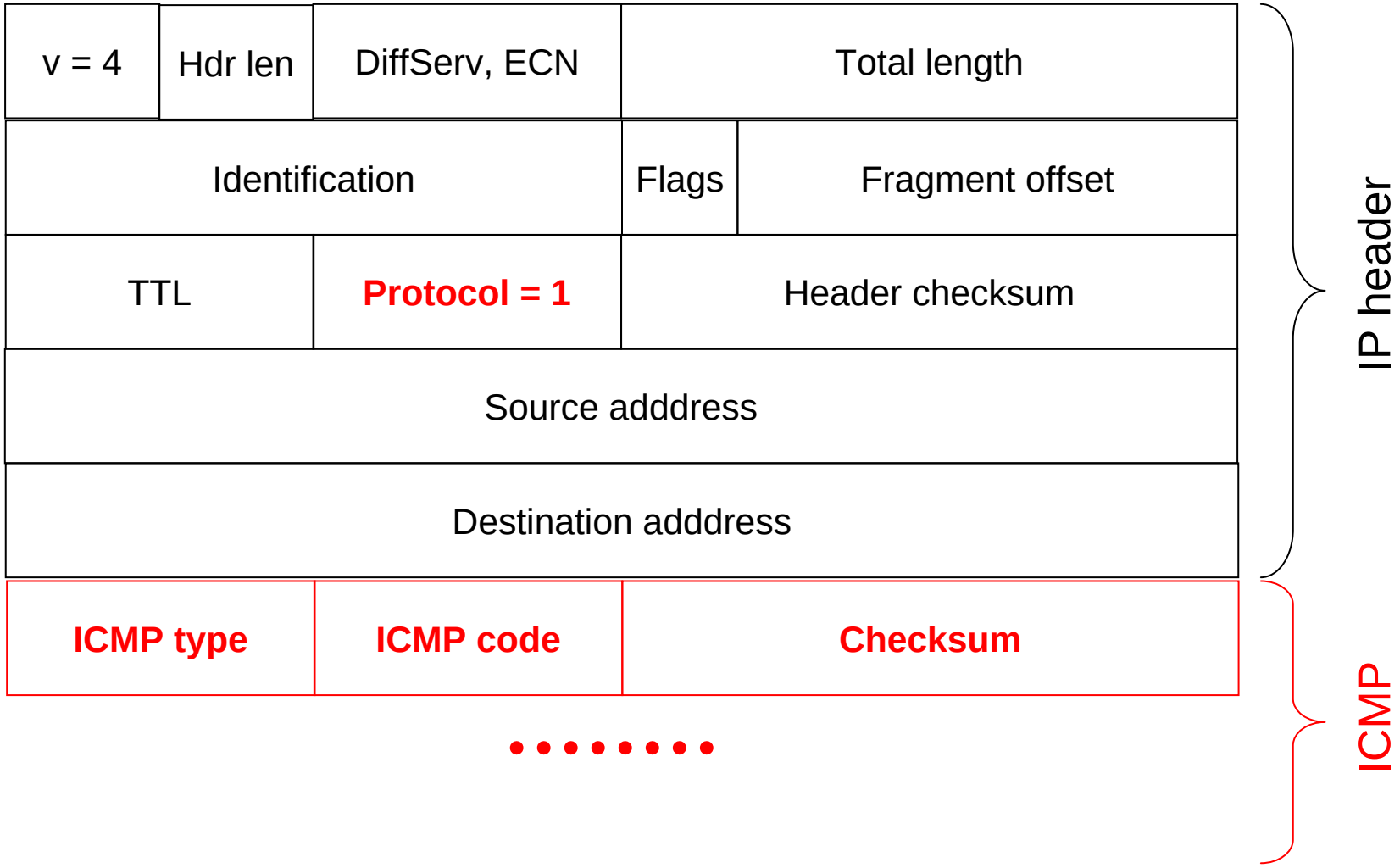
- Options and fragmentation:
  - Some options must be copied into each fragment, e.g. source route options.
  - Others remain only in the first fragment, e.g. timestamp option.
- Loose versus strict source routing:
  - Strict source routes enumerate all intermediary routers. If one of them is not directly reachable, the packet is discarded.
  - Loose source routes give a list of routers that must be visited, but ‚normal‘ routing may be used to reach them.
- IPv4 options are rarely used in practice:
  - Source routing is considered as security problem.
  - Many terminals drop packets with options.
  - But options (esp. new options introduced after RFC 791) could be beneficial.

Internet Control Message Protocol (ICMP)

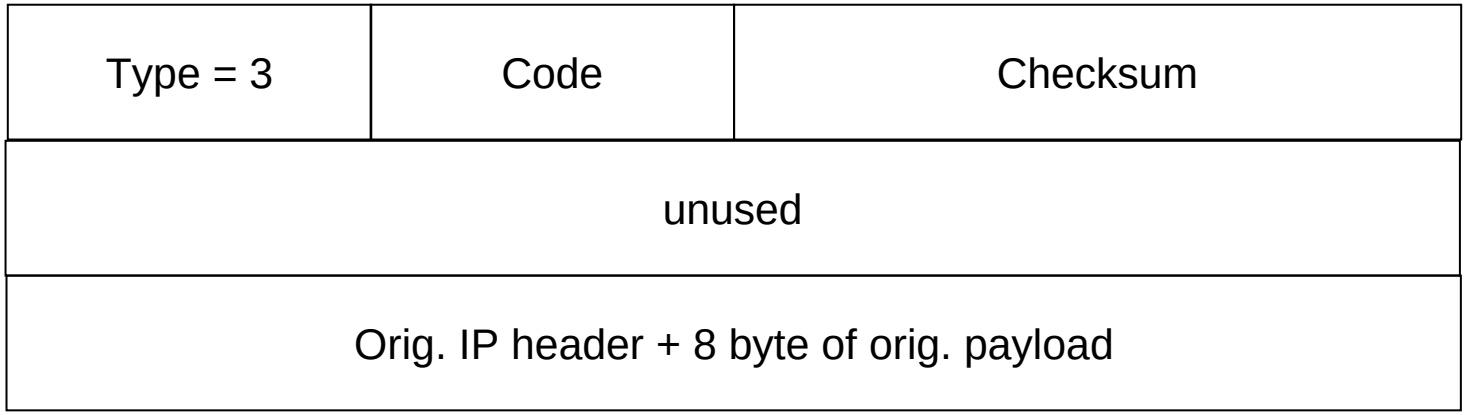
- Remember Path MTU discovery:
  - Send a packet with ‚don‘t fragment‘ flag set.
  - Router notifies the sender if packet size exceeds the MTU of the next layer 2 network.
  - Sender then reduces the packet size accordingly.
- Such notifications are done by ICMP.

- There exist various ICMP message types, some with (sub-) codes:
  - 0 = Echo Reply
  - 3 = Destination Unreachable (see codes for details)
  - 4 = Source Quench
  - 5 = Redirect
  - 8 = Echo Request
  - 9 = Router Advertisment
  - 10 = Router Solicitation
  - 11 = Time Exceeded

ICMP Header


v = 4
Hdr len
DiffServ, ECN
Total length
Identification
Flags
Fragment offset
TTL
Protocol = 1
Header checksum
Source adddress
Destination adddress
ICMP type
ICMP code
Checksum
IP header
ICMP

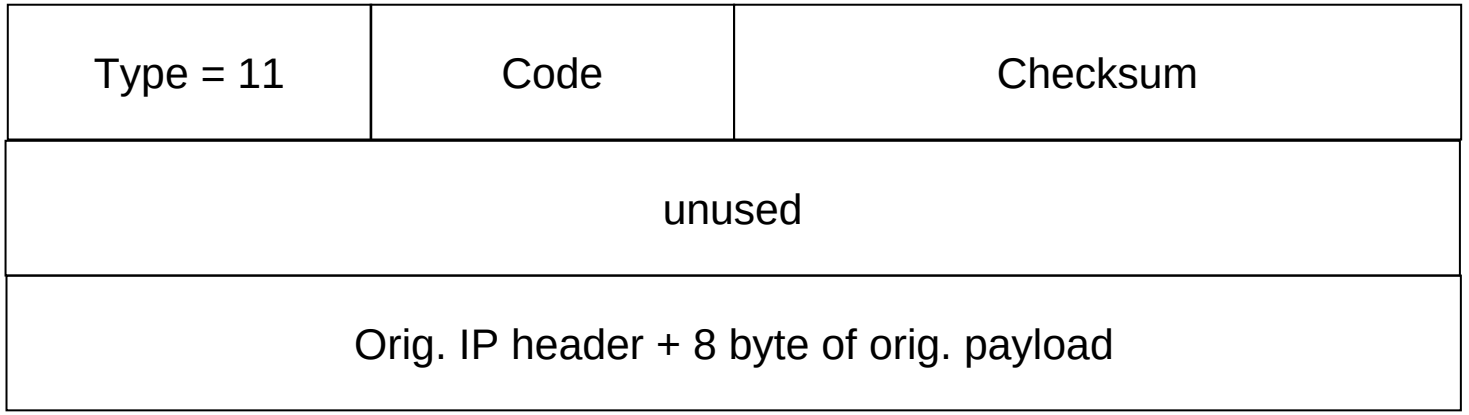
ICMP Destination Unreachable



Code

- 0 = net unreachable;
- 1 = host unreachable;
- 2 = protocol unreachable;
- 3 = port unreachable;
- 4 = fragmentation needed and DF set;
- 5 = source route failed.

ICMP Time Exceeded



Code

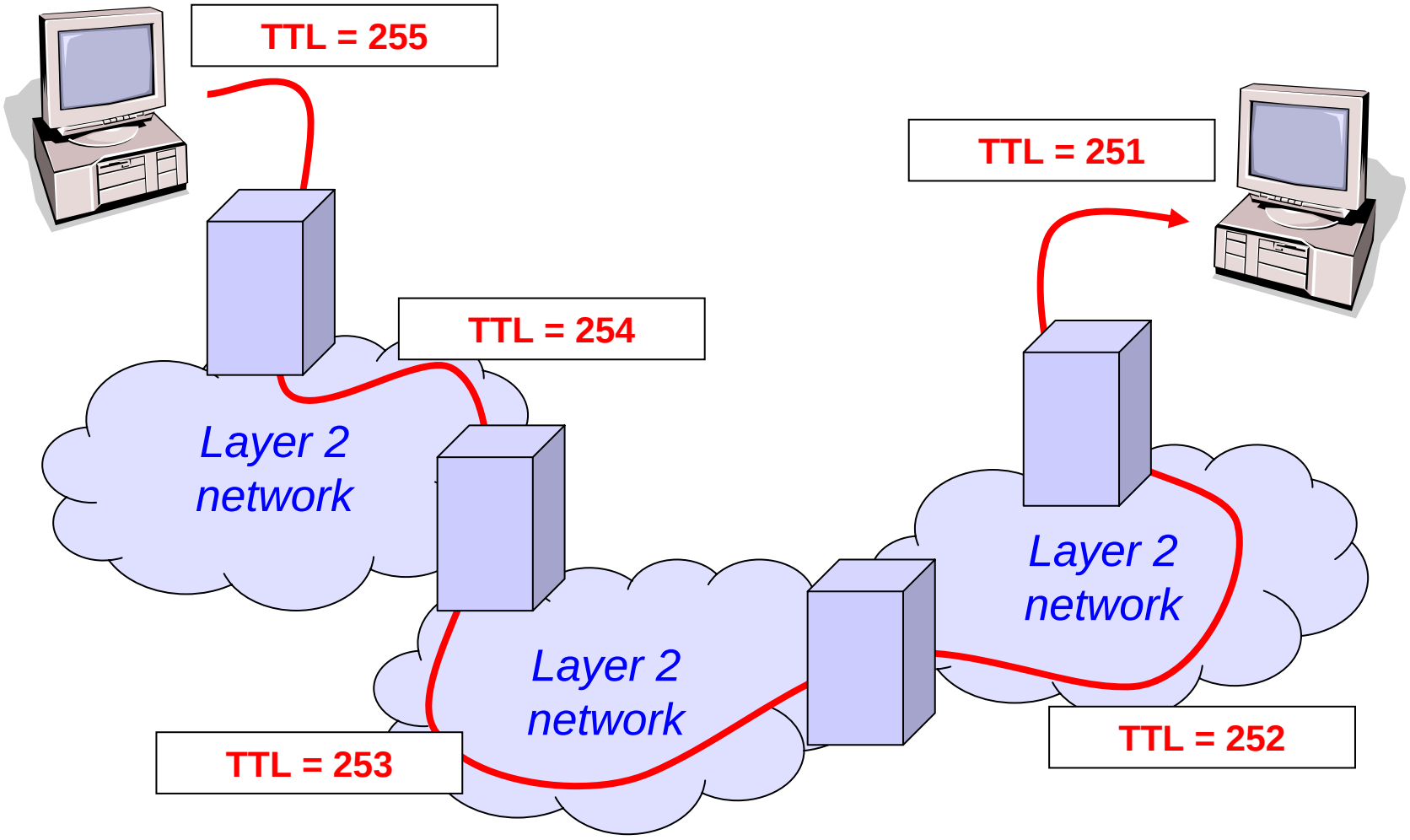
0 = time to live exceeded in transit;

1 = fragment reassembly time exceeded.

Note: ***These messages are not sent if the original packet was an ICMP message.*** Otherwise, there could be an endless ping-pong of ICMP messages.

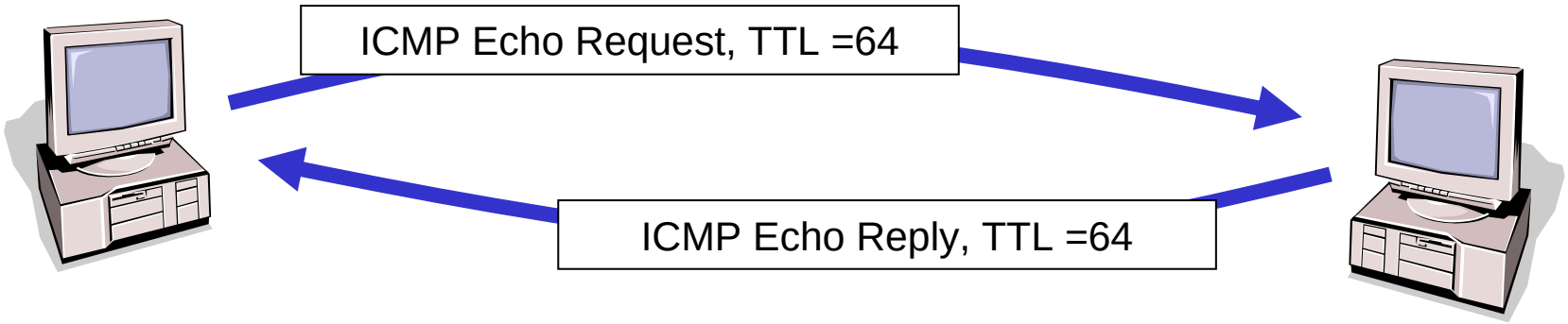
Example: Consider a routing loop, i.e. packets are forwarded without ever arriving at the destination. If the ICMP TTL exceeded message was also caught in this loop, the message would never die out.

Time to Live Example



ICMP Echo Request & Reply

Type = 8 or 0	Code = 0	Checksum
Identification		Sequence number
Data …		




```
PING phoenix.net.in.tum.de (131.159.14.1) 56(84) bytes of data.
64 bytes from phoenix.net.in.tum.de (131.159.14.1): icmp_seq=1 ttl=50 time=5.51 ms
64 bytes from phoenix.net.in.tum.de (131.159.14.1): icmp_seq=2 ttl=50 time=5.37 ms
64 bytes from phoenix.net.in.tum.de (131.159.14.1): icmp_seq=3 ttl=50 time=5.39 ms
64 bytes from phoenix.net.in.tum.de (131.159.14.1): icmp_seq=4 ttl=50 time=5.41 ms
```

Questions

- Read the Wikipedia article on ‚Switch (Computertechnik)‘. – Here, the German version is better than the English.
  - Summarize the article and explain the problems in switched networks! How can these problems be solved?
  - Read „Interconnections: Bridges, Routers, Switches, and Internetworking Protocols” by Radia Perlman if you are interested in further details.
- Explain the design rationale for the IPv4 header fields!
- Explain how to build the Unix applications ‘ping’ and ‘traceroute’.
- Describe how to build an application that determines the Path MTU!

Hard questions (for 2nd question you need to know what a NAT is):

- What happened if you receive a ‘fragment reassembly time exceeded’?
- Why are source routed IP packets considered harmful? – Spoiler: Try to send a packet to a host behind a NAT!

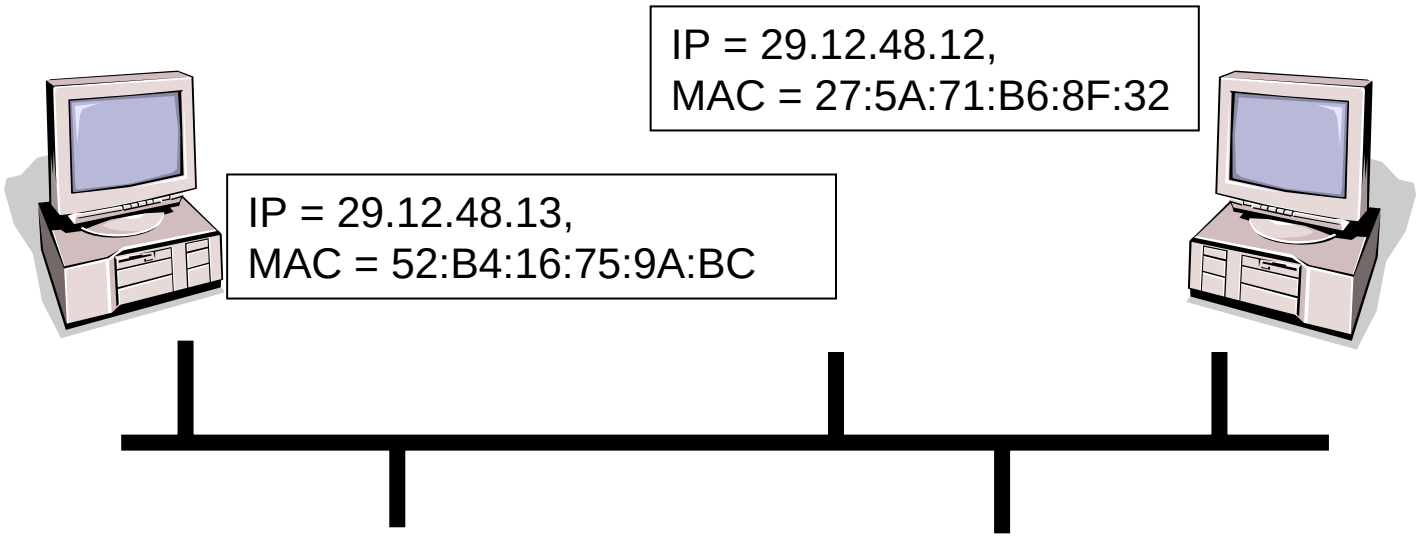
1. Fragmenting IP Packets

2. Options in the IP header

3. Global versus Local Addresses

4. Routing

MAC and IP Addresses



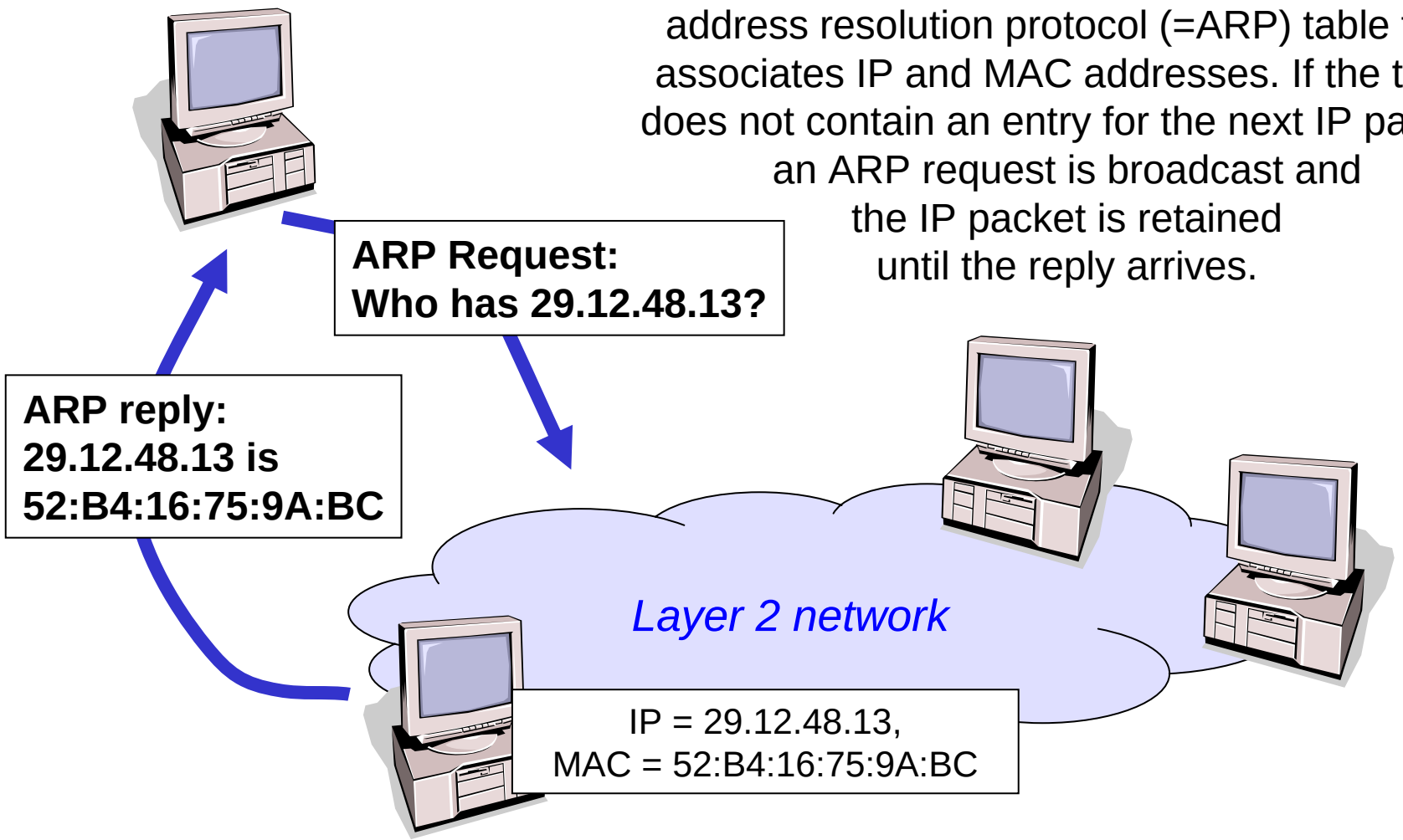
Data link layer has own addresses, e.g. IEEE 802 addresses.

- IEEE 802 addresses are used with Ethernet, Token Ring, Wireless LAN, etc.
- These so called MAC addresses have typically unrelated to IP addresses.
- IEEE 802 addresses are 48 bit and assigned by the hardware manufacturer.
- IPv4 addresses are 32 bit and assigned by the network provider.

How does a station know what to put into the MAC header?

The Address Resolution Protocol (=ARP)

Each internet host maintains a so called address resolution protocol (=ARP) table that associates IP and MAC addresses. If the table does not contain an entry for the next IP packet, an ARP request is broadcast and the IP packet is retained until the reply arrives.

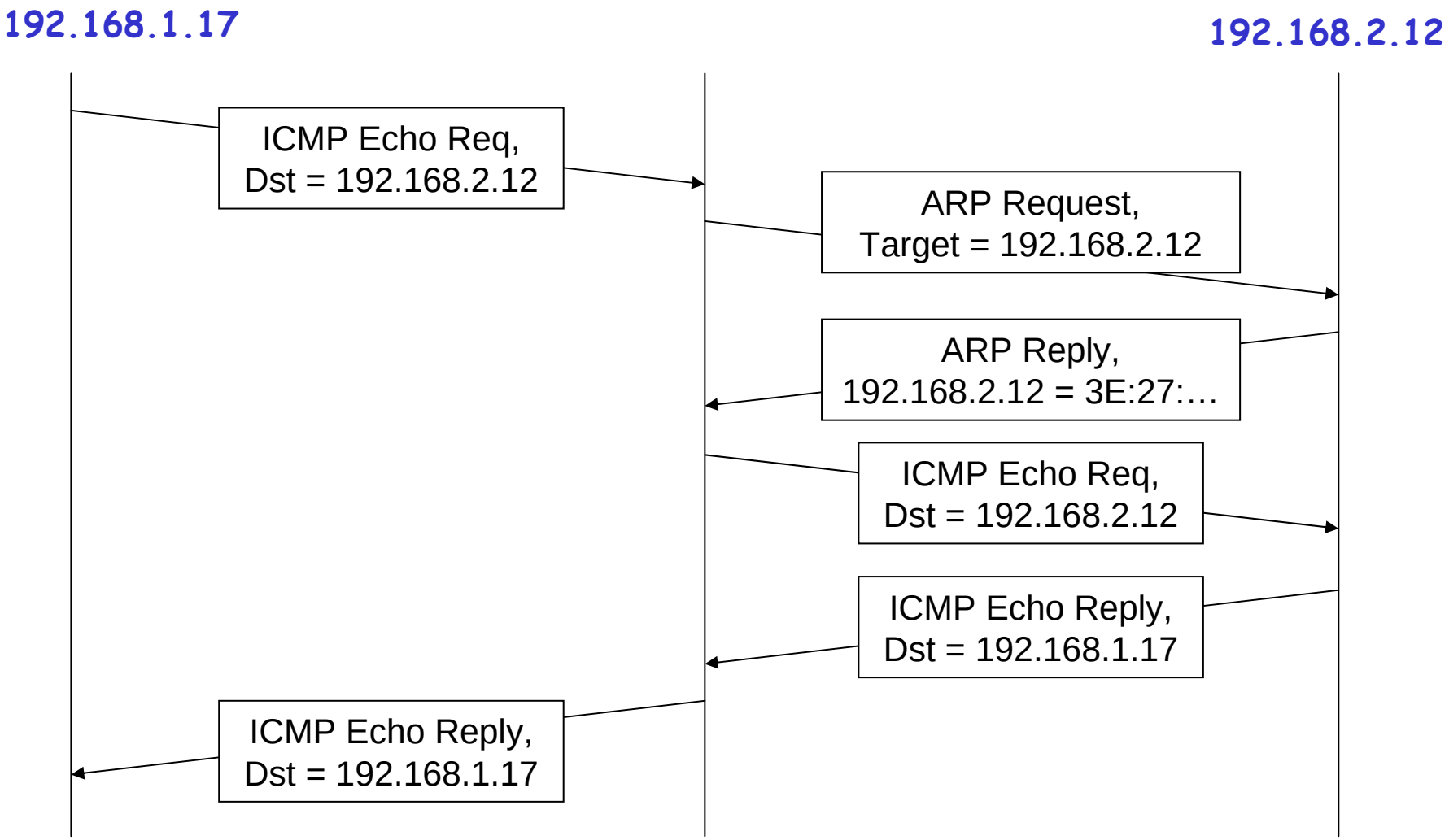


ARP Packet Format

Hardware type e.g. Ethernet = 1		Protocol type, e.g. IPv4 = 2048
Hardware length e.g. Ethernet = 6	Protocol length e.g. IPv4 = 4	Operation 1 = request, 2 = reply
Sender hardware address		
Sender protocol address		
Target hardware address (set to zero in request)		
Target protocol address		

Note: Of course, ARP is carried in a data link layer datagram such as an Ethernet frame!

ARP Example


192.168.1.17
192.168.2.12
ICMP Echo Req,
Dst = 192.168.2.12
ARP Request,
Target = 192.168.2.12
ARP Reply,
192.168.2.12 = 3E:27:…
ICMP Echo Req,
Dst = 192.168.2.12
ICMP Echo Reply,
Dst = 192.168.1.17
ICMP Echo Reply,
Dst = 192.168.1.17

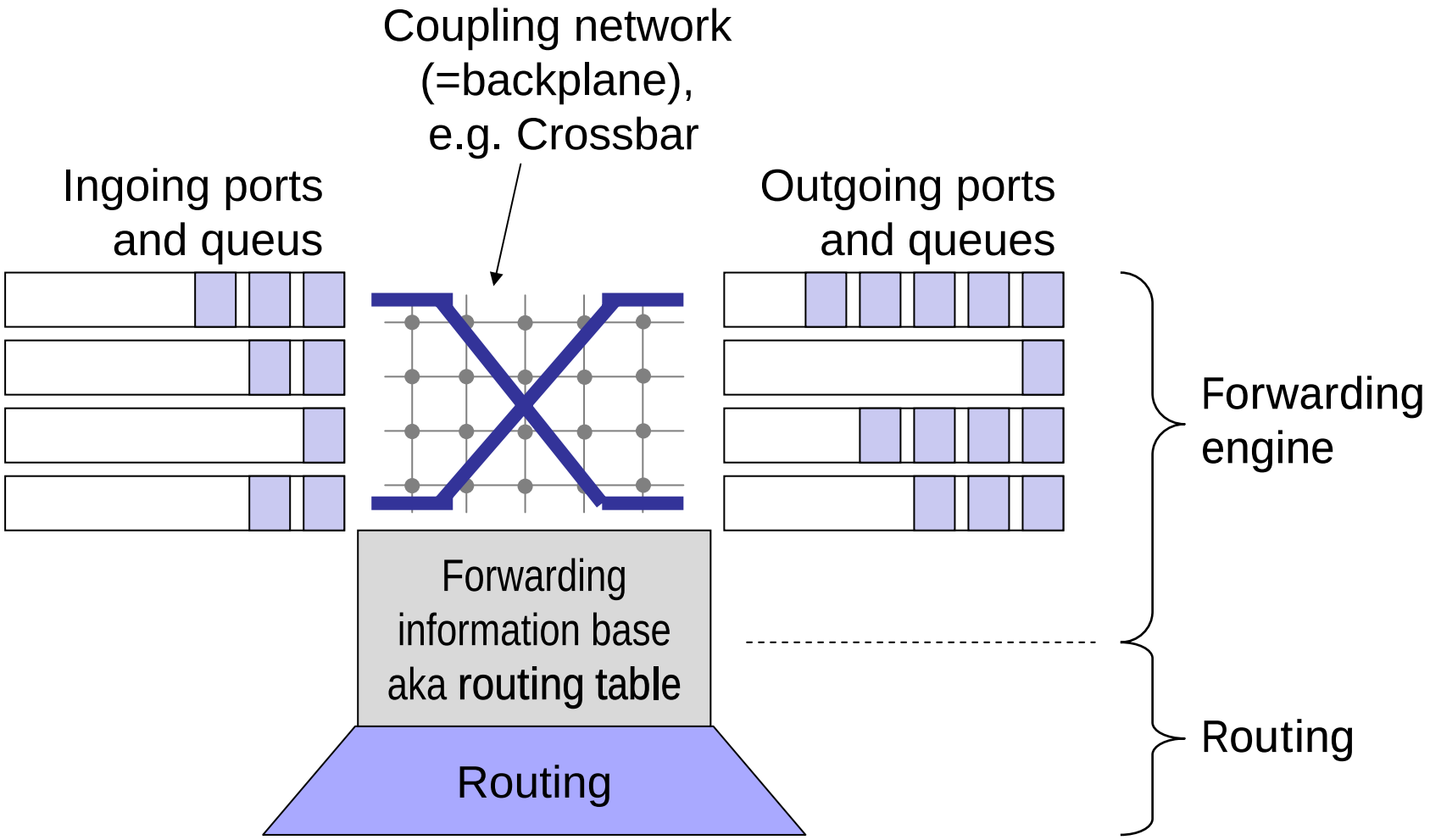
1. Fragmenting IP Packets

2. Options in the IP header

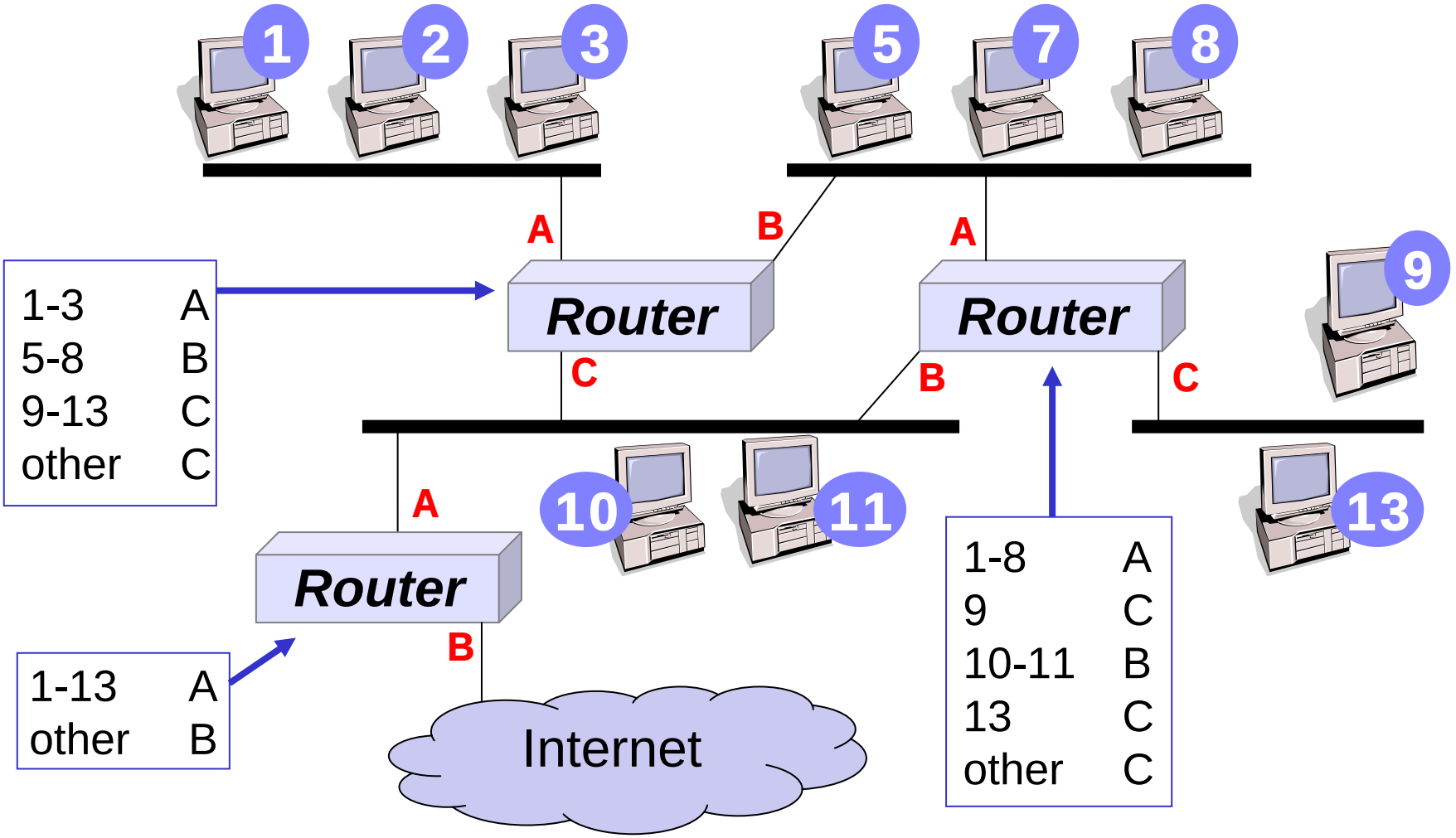
3. Global versus Local Addresses

4. Routing

Router – An Overview

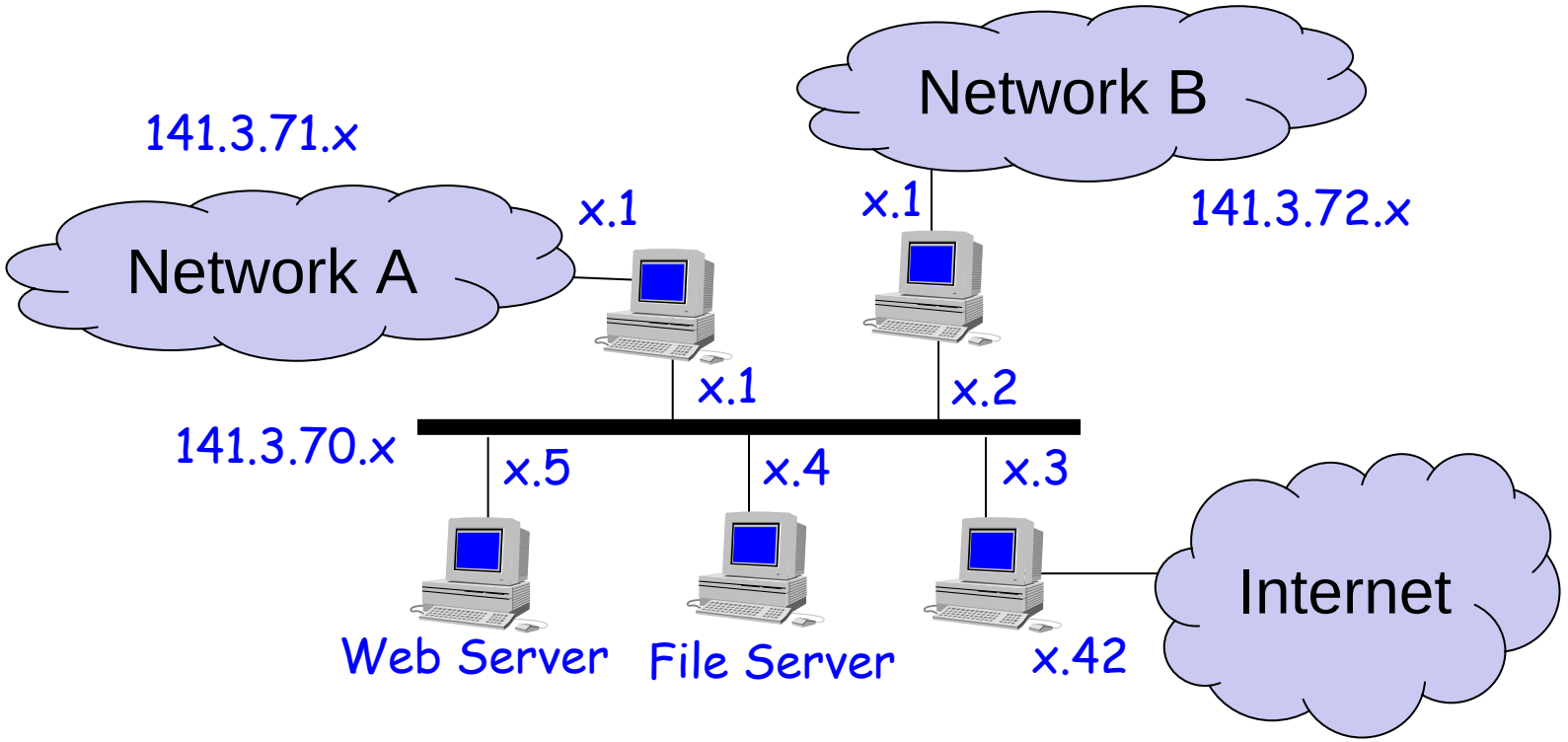


Routing Tables



Routing Tables and IP Addresses (1)

1. Routing tables do not (only) say to which of the attached networks the packet shall be forwarded, but also to which IP address there (router or terminal host). – Why?
2. Routers which are typically attached to more than one network, have a different IP address in each of these networks. – Why?



Routing Tables and IP Addresses (2)

- Answers:
  - Typically, layer 2 networks contain more that one other IP router / host. We need to say which one is in charge of the packet.
  - IP addresses in the same network have a common prefix, so a router must have different addresses to match the prefixes of the different networks.
- Prefix matching is a core principle of the Internet‘s scalability:
  - Since all hosts in a network have the same prefix, one entry in the routing table suffices for all hosts with the same prefix.
  - Normally, the IP addresses are assigned so that the prefixes can be well aggregated.
- Originally (RFC 791) each IP address consisted of a well distinguished network and local address:
  - Class A (most significant bit = 0), local address has 24 bit
  - Class B (most significant bits = 10), local address has 16 bit
  - Class C (most significant bits = 110), local address has 8 bit

Routing Overview

- Routing has to fill the forwarding/routing information base (FIB or RIB).
- Various possible approaches
  - Static routing, RIB filled manually, e.g. by site administrator
  - Centralized routing, a central service (manual or automted) fills the RIBs of all routers in the network
  - Decentralized, adaptive ($\rightarrow$ self-organized) routing: A distributed algorithm fills the RIBs
- Various decentralized, adaptive routing algorithms
  - Stable and adaptive
  - Simple and low overhead
  - … hard to achieve all desirable properties!
  - Sometimes manual control required

Static Routing

- Routing tables are simply set up by the network administrator
- Well suited for stub networks which have only one router connecting the network to the rest of the world.

Example:

- Router is attached to two networks: 141.3.71.x via interface ‚eth0‘ and 192.168.1.x via interface ‚eth1‘
- Hosts in those networks are reached directly (i.e. no other router involved).
- All other host shall be sent via router 141.3.71.126

Ziel	Router	Genmask	Metric	Iface
141.3.71.0	0.0.0.0	255.255.255.0	0	eth0
192.168.1.0	0.0.0.0	255.255.255.0	0	eth1
0.0.0.0	141.3.71.126	0.0.0.0	0	eth0

Flooding

- Very simple and robust, i.e. can adapt to quickly changing conditions.
- But creates very much overhead in the network.
- Overview:
  - Forward (copies of) each packet on all interfaces, except the one from which the packet arrived.
  - Simple flooding suffers from loops!
  - Must have a TTL limit for forwarding!
- Optimizations:
  - Use unique packet IDs to detect loops
  - Use source routes to detect loops
- Variants:
  - Selective flooding: Forward (copies of) packets only on some interfaces.
  - Random Walk: Pick only one interface (no copies any more!)

Hot Potato

- Jeder Knoten versucht, eingehende Pakete so schnell wie möglich weiterzuleiten, d.h. Übertragung auf Leitung mit der kürzesten Warteschlange
- Probleme:
  - Ziel wird möglicherweise nicht erreicht
  - Übertragungsweg mit hoher Wahrscheinlichkeit suboptimal (Schleifen, ...)
- Varianten:
  - Weiterleitung nie auf der Übertragungsleitung, auf der Paket eintraf
  - Kombination mit statischem Routing
    - Auswahl der besten Übertragungsleitung nach statischem Verfahren, solange deren Warteschlangenlänge unter einer bestimmten Schwelle bleibt
    - Auswahl der Übertragungsleitung mit kürzester Warteschlange, falls deren statisches Gewicht nicht zu niedrig ist

Backward Learning

- Daten im Paket
  - Identifikation des Quellknotens
  - Ein Zähler, der mit jeder zurückgelegten Teilstrecke (=„Hop“) um 1 erhöht wird.
- Beispiel
  - Falls z.B. bei einem Knoten K auf der Übertragungsleitung k ein Paket mit Zähler = 4 vom Ursprungsknoten H eintrifft, so weiß der Knoten K, dass er den Knoten H über die Leitung k in 4 hops erreichen kann.
  - Falls Knoten Ks bisheriger geschätzter optimaler Weg zu Knoten H mehr als 4 hops beträgt, so aktualisiert Knoten K seine Routingtabelle mit dem neuen und jetzt besseren Weg.
- Nachteile
  - Nur Änderungen zum Besseren werden zur Kenntnis genommen
  - Ausfälle oder Überlastung von Übertragungsleitungen werden nicht weitergemeldet.
- Folge
  - Knoten müssen periodisch alle Informationen vergessen und wieder von neuem aufsetzen.
- Problem
  - Während der Lernperiode ist die Wegewahl nicht optimal.
  - Bei häufigem Neubeginn nehmen viele Pakete Wege unbekannter Qualität.
  - Bei seltenem Neubeginn ergibt sich ein schlechtes Anpassungsverfahren

Verteiltes adaptives Routing (1)

- Knoten tauschen periodisch Routing-Informationen mit ihren Nachbarn aus.
- Typischerweise unterhält jeder Knoten eine Routing-Tabelle, die für jeden anderen Knoten im Netz einen Eintrag enthält. Hierzu zählen
  - Bevorzugte Übertragungsleitung für diesen Knoten
  - Schätzung über Zeit oder Entfernung zu diesen Knoten:
    - Anzahl Hops,
    - Geschätzte Verzögerung in Millisekunden sowie
    - Geschätzte totale Anzahl von Paketen, die entlang des Weges warten.
  - Die Schätzungen werden gewonnen aus
    - Zeit oder Entfernung zu den Nachbarn (z.B. aus speziellen Echo Paketen mit Zeitstempeln)
    - Schätzungen der Nachbarn
- Informationsaustausch mit den Nachbarn kann in zwei Varianten geschehen
  - Synchroner Austausch in bestimmten Update-Intervallen
  - Asynchroner Austausch bei signifikanten Änderungen

Verteiltes adaptives Routing (2)

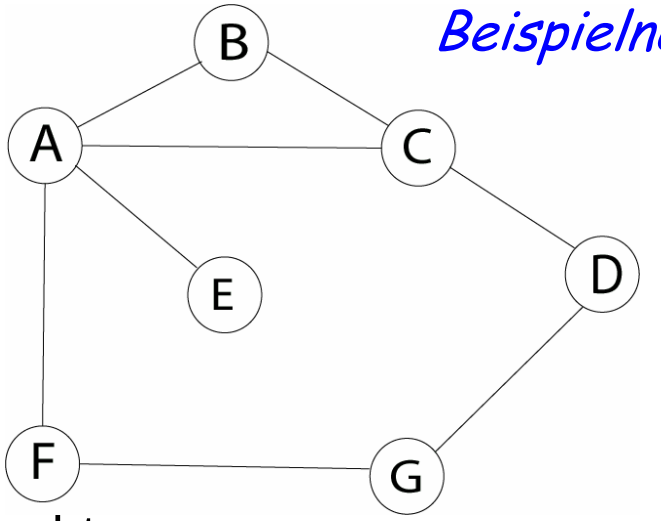
Im praktischen Einsatz: verteilte adaptive Routingalgorithmen. Dabei werden zwei grundlegende Algorithmen unterschieden:

- ***Distanz-Vektor-Algorithmen***
  - Jedes System kennt die Distanz zu allen anderen Systemen im Netz
  - Aktuellen Distanzen werden zwischen den Nachbarn ausgetauscht
  - Aber: kürzerer langsamerer Weg wird längerem schnelleren Weg vorgezogen
  - Beispiele: Routing Information Protocol (RIP), Distance-Vector-Routing-Protocol (DVRP)

- ***Link-State-Algorithmen***
  - Unterschiedliche Routing-Metriken möglich
  - berücksichtigt die aktuellen Zustände der Netzanschlüsse
  - jeder Router kennt die komplette Netztopologie und berechnet auf dieser Basis seine Routinginformation
  - Link-State-Algorithmen konvergieren im allgemeinen schneller als Distanz-Vektor-Algorithmen, d.h. besser geeignet für größere Netze
  - Beispiele: Open Shortest Path First (OSPF), Intra-Domain Intermediate System to Intermediate System Routing Protocol (IS-IS)

Distanz Vektor Routing

- Einfaches adaptives verteiltes Verfahren
- Als „Routing Information Protocol“ (RIP) anfangs im Internet verwendet
- Jeder Router speichert eine Tabelle mit der besten Ausgangsleitung und zugehöriger Entfernung (z.B. Anzahl Hops, Verzögerung) zu jedem Ziel

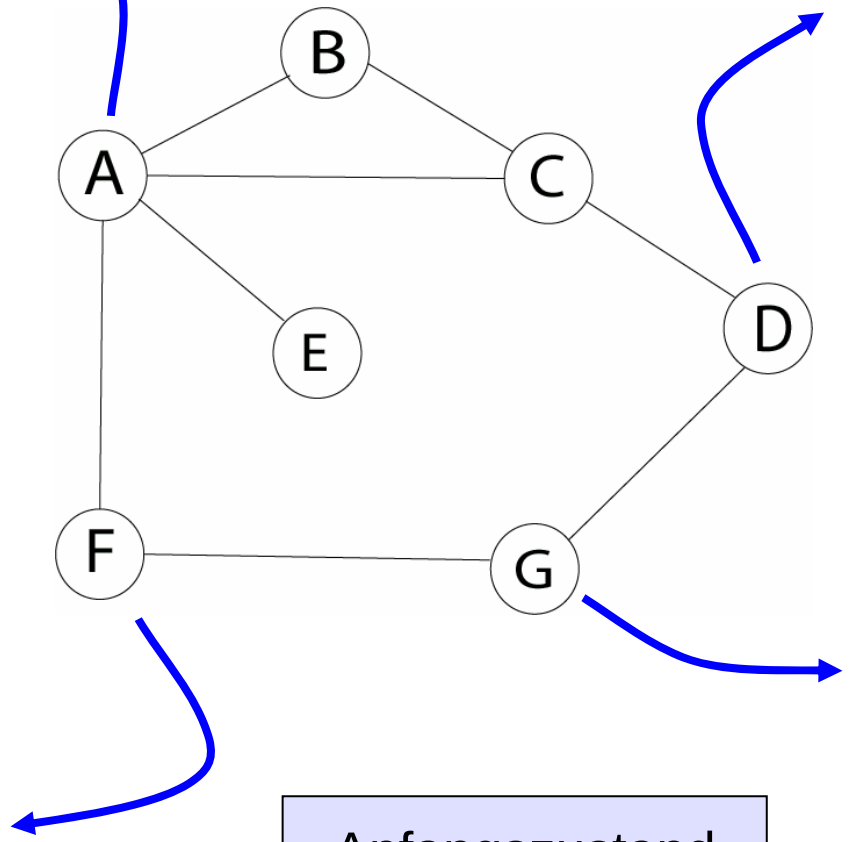


- Tabellen werden mit Nachbarn ausgetauscht
- Entfernung plus 1 ergibt Entfernung über den jeweiligen Nachbarn
- Übernehme Eintrag falls besser als bisheriger Eintrag
- Praktisches Problem: langsame Konvergenz zu einem konsistenten Zustand und „count-to-infinity“ -Problematik

Beispiel: Distanz Vektor Routing (1)

Ziel	Hops	Next
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

Ziel	Hops	Next
A	1	A
B	∞	-
C	∞	-
D	∞	-
E	∞	-
G	1	G



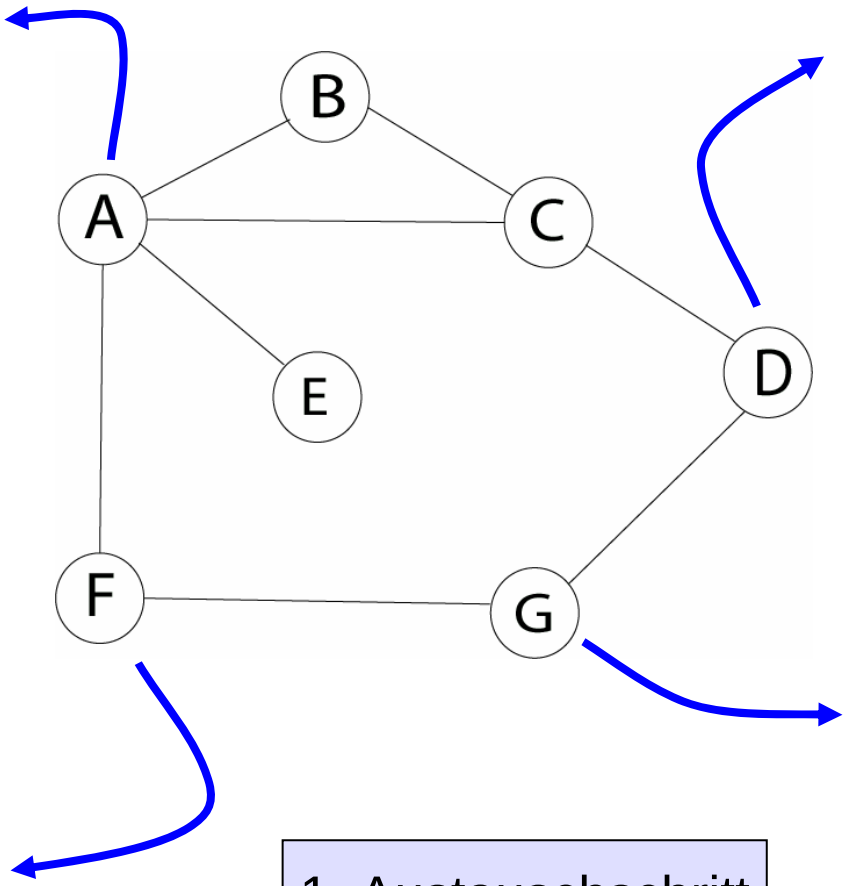
Ziel	Hops	Next
A	∞	-
B	∞	-
C	1	C
E	∞	-
F	∞	-
G	1	G

Ziel	Hops	Next
A	∞	-
B	∞	-
C	∞	-
D	1	D
E	∞	-
F	1	F

Beispiel: Distanz Vektor Routing (2)

Ziel	Hops	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Ziel	Hops	Next
A	1	A
B	2	A
C	2	A
D	2	G
E	2	A
G	1	G



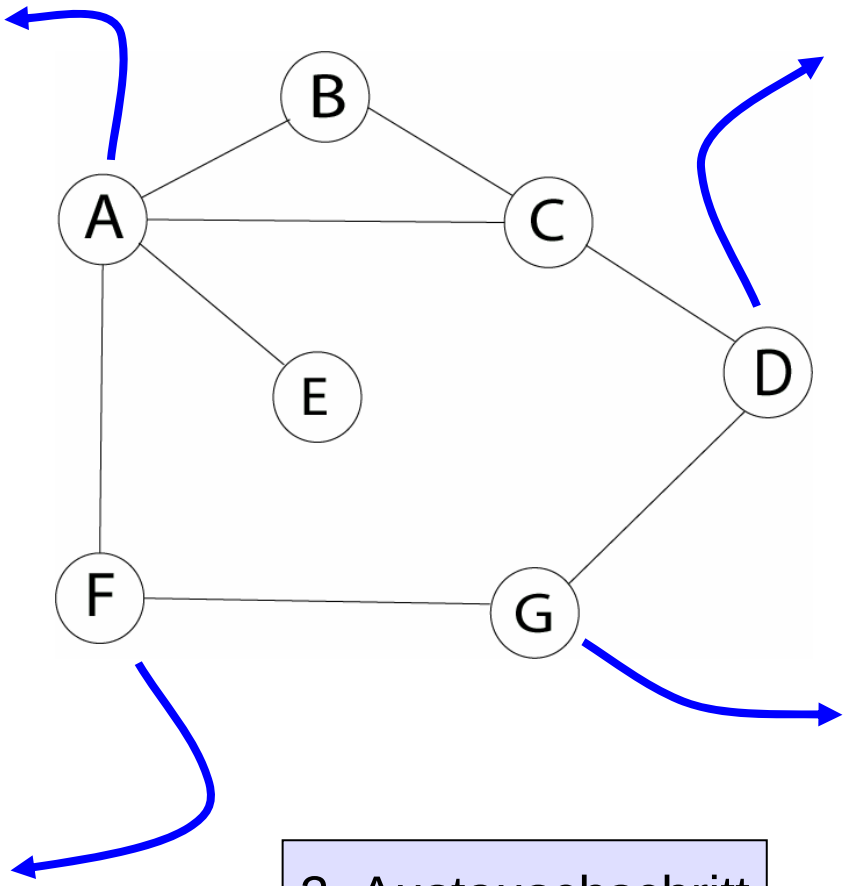
Ziel	Hops	Next
A	2	C
B	2	C
C	1	C
E	∞	-
F	2	G
G	1	G

Ziel	Hops	Next
A	2	F
B	∞	-
C	2	D
D	1	D
E	∞	-
F	1	F

Beispiel: Distanz Vektor Routing (3)

Ziel	Hops	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Ziel	Hops	Next
A	1	A
B	2	A
C	2	A
D	2	G
E	2	A
G	1	G



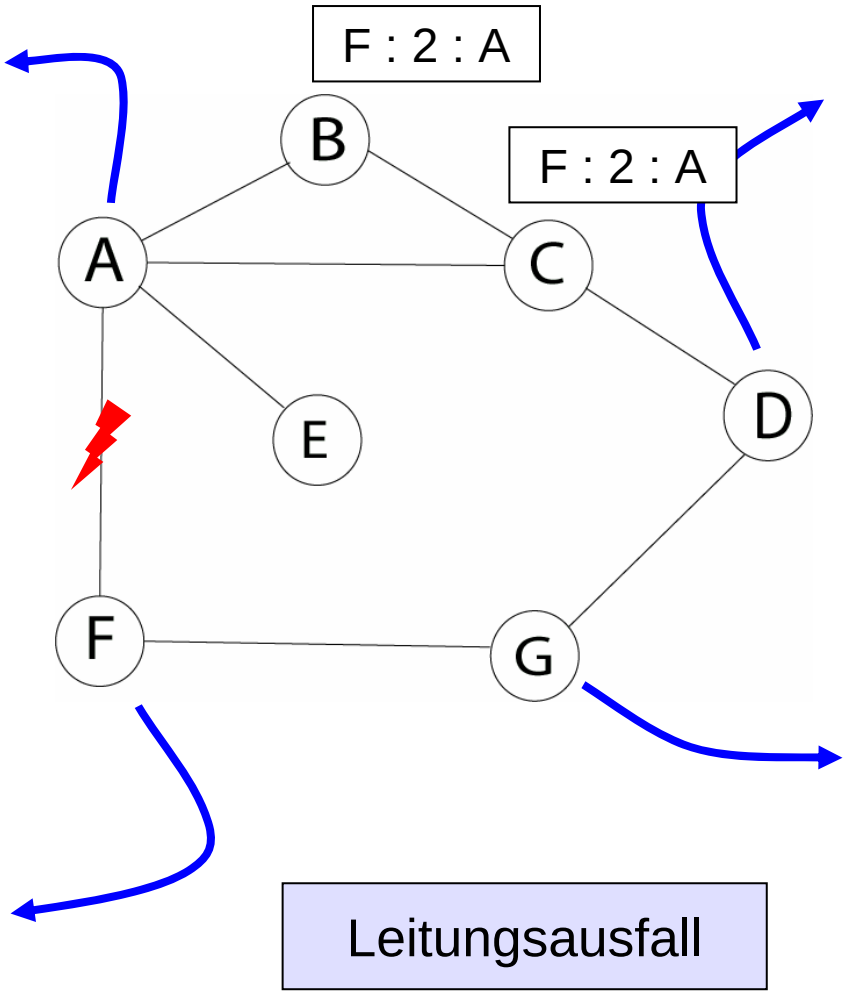
Ziel	Hops	Next
A	2	C
B	2	C
C	1	C
E	3	C
F	2	G
G	1	G

Ziel	Hops	Next
A	2	F
B	3	D
C	2	D
D	1	D
E	3	F
F	1	F

Beispiel: Distanz Vektor Routing (4)

Ziel	Hops	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	∞	-
G	∞	-

Ziel	Hops	Next
A	∞	-
B	∞	-
C	∞	-
D	2	G
E	∞	-
G	1	G



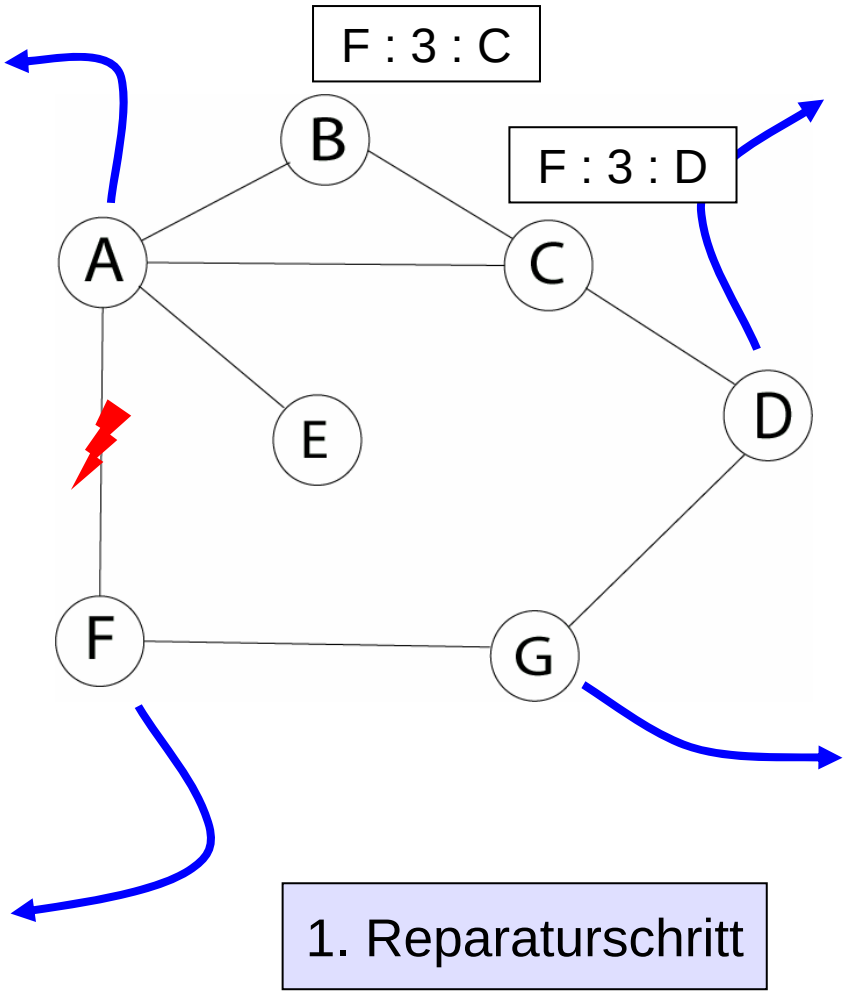
Ziel	Hops	Next
A	2	C
B	2	C
C	1	C
E	3	C
F	2	G
G	1	G

Ziel	Hops	Next
A	2	F
B	3	D
C	2	D
D	1	D
E	3	F
F	1	F

Beispiel: Distanz Vektor Routing (5)

Ziel	Hops	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	**3**	**B**
G	3	C

Ziel	Hops	Next
A	**3**	**G**
B	4	G
C	3	G
D	2	G
E	**4**	**G**
G	1	G



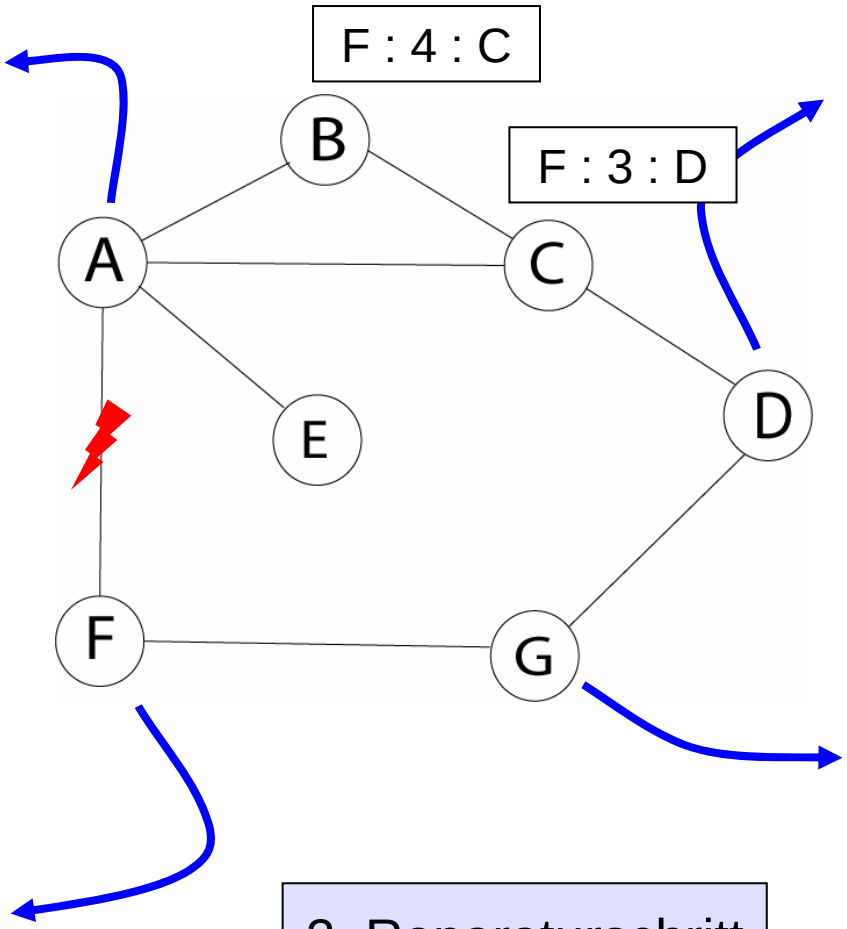
Ziel	Hops	Next
A	2	C
B	2	C
C	1	C
E	3	C
F	2	G
G	1	G

Ziel	Hops	Next
A	3	D
B	3	D
C	2	D
D	1	D
E	4	D
F	1	F

Beispiel: Distanz Vektor Routing (6)

Ziel	Hops	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	**4**	**B**
G	3	C

Ziel	Hops	Next
A	4	G
B	4	G
C	3	G
D	2	G
E	5	G
G	1	G



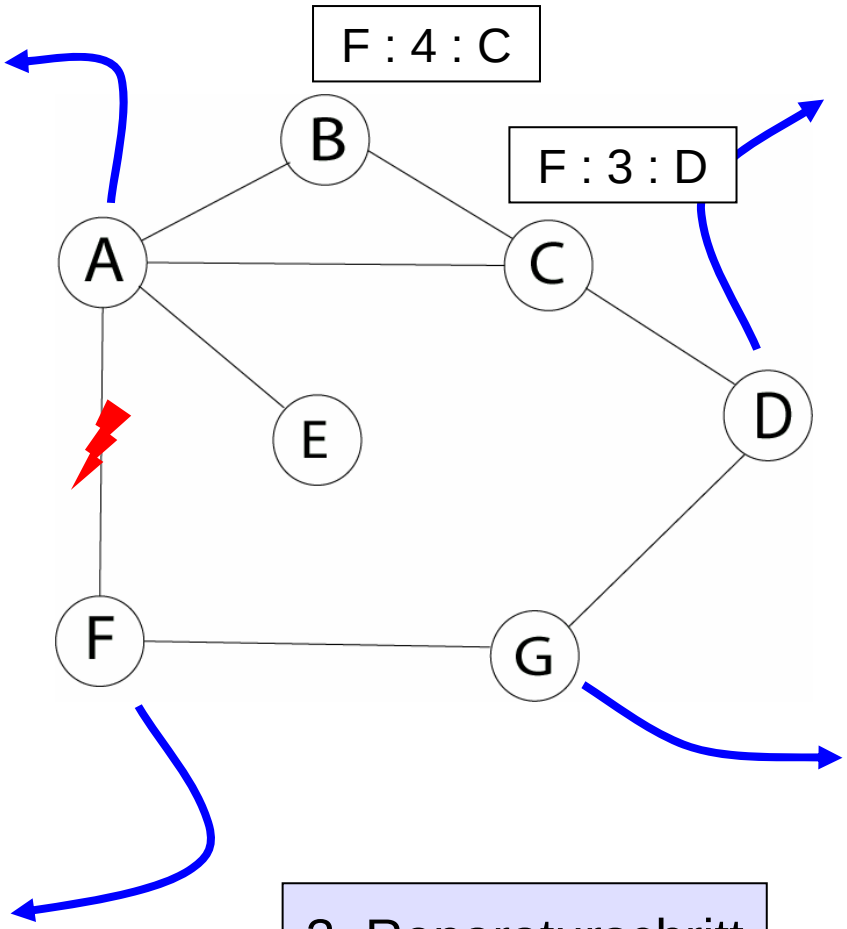
Ziel	Hops	Next
A	2	C
B	2	C
C	1	C
E	3	C
F	2	G
G	1	G

Ziel	Hops	Next
A	3	D
B	3	D
C	2	D
D	1	D
E	4	D
F	1	F

Beispiel: Distanz Vektor Routing (7)

Ziel	Hops	Next
B	1	B
C	1	C
D	2	C
E	1	E
F	4	C
G	3	C

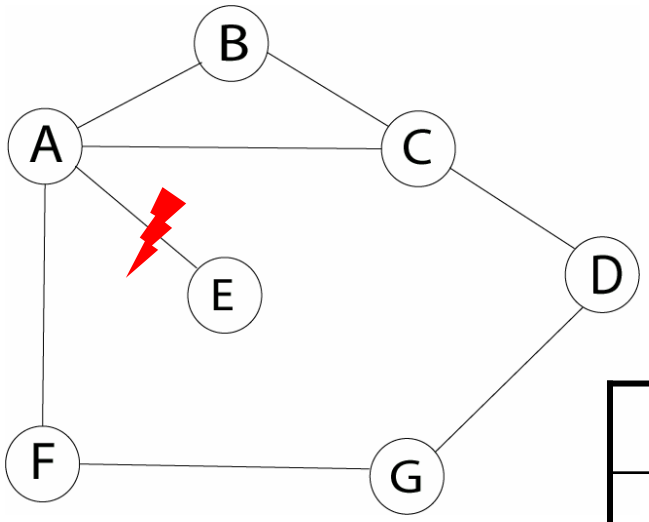
Ziel	Hops	Next
A	4	G
B	4	G
C	3	G
D	2	G
E	5	G
G	1	G



Ziel	Hops	Next
A	2	C
B	2	C
C	1	C
E	3	C
F	2	G
G	1	G

Ziel	Hops	Next
A	3	D
B	3	D
C	2	D
D	1	D
E	4	D
F	1	F

Count-to-Infinity Problem



Beispielnetz

- Unerreichbarkeit von Knoten wird nur durch ein allmähliches Hochzählen der Entfernung bemerkt

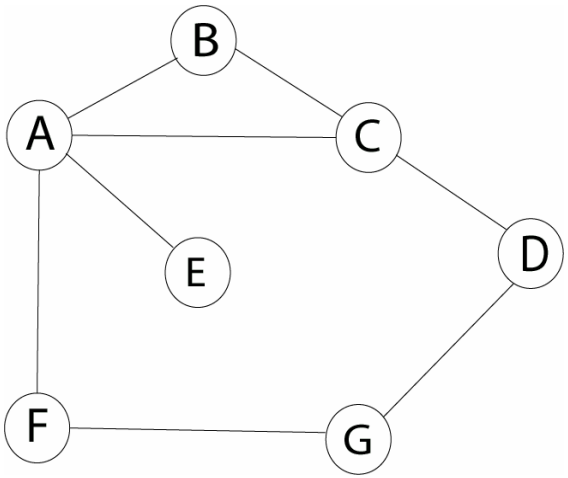
	A	B	C	D	F	G
	1 : E	2 : A	2 : A	3 : C	2 : A	3 : F
Ausfall	∞	2 : A	2 : A	3 : C	2 : A	3 : F
1. Schritt	3 : B	3 : C	3 : B	3 : C	4 : G	3 : F
2. Schritt	4 : B	4 : A	4 : A	4 : C	4 : A	5 : F
3. Schritt	5 : B	5 : A	5 : A	5 : C	5 : A	5 : F
4. Schritt	6 : B	6 : A	6 : A	6 : C	6 : A	6 : F
5. Schritt	7 : B	7 : A	7 : A	7 : C	7 : A	7 : F

Max. Netzdurchmesser = Infinity

Typischer Wert: 16

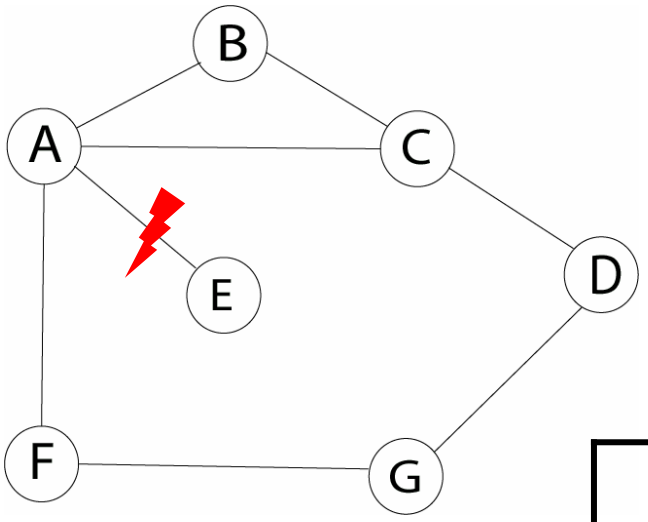
Split Horizon & Poison Reverse

- Distanz Vektor Protokolle konvergieren langsam, da sich Informationen nur Schritt für Schritt verbreiten
- Aber: Werden die Tabellen „einfach so“ übernommen, verbreiten sich sogar sinnlose Informationen (vgl. Beispiel)
- Lösung: Gebe nur Informationen weiter, die für den Nachbarn sinnvoll sein könnten:
  - ***Split Horizon:*** Nachbar X erhält nur Tabelleneinträge die nicht über X selbst als Next Hop führen.
  - ***Poison Reverse:*** Für Nachbar X werden alle Einträge, die über X selbst führen auf „Unendlich“ gesetzt.



Ziel	Hops	Next	Weiter-leiten an…
B	1	B	C, E, F
C	1	C	B, E, F
D	2	C	B, E, F
E	1	E	B, C, F
F	1	F	B, C, E
G	2	F	B, C, E

Beispiel: Poison Reverse



Beispielnetz

- Mit Poison Reverse verbreitet sich die Unerreichbarkeit von Knoten vergleichsweise rasch.
- Veränderungen im Abstand n Hops werden i.d.R. nach n Schritten erkannt.

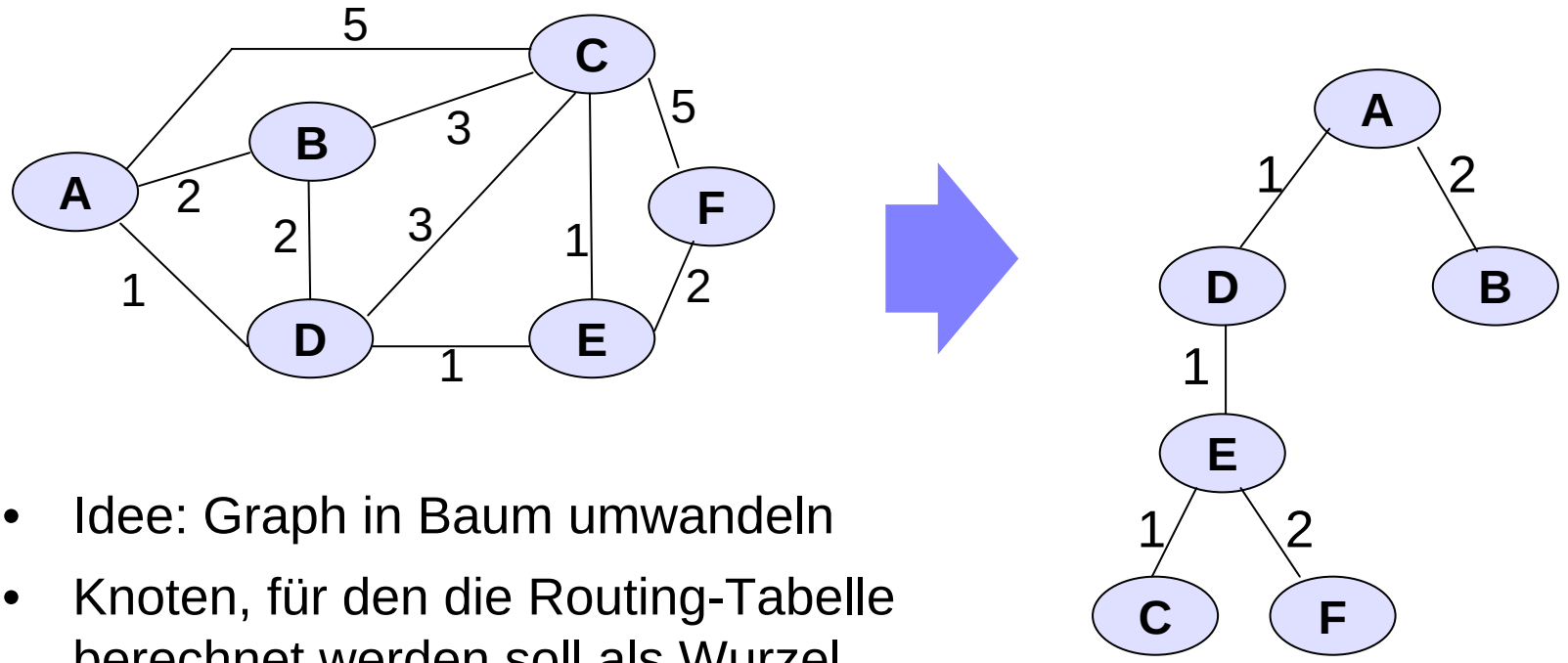
	A	B	C	D	F	G
	1 : E	2 : A	2 : A	3 : C	2 : A	3 : F
Ausfall	∞	2 : A	2 : A	3 : C	2 : A	3 : F
1. Schritt	∞	3 : C	3 : B	3 : C	∞	3 : F
2. Schritt	∞	∞	∞	4 : C	∞	4 : D
3. Schritt	∞	∞	∞	∞	∞	5 : D
4. Schritt	∞	∞	∞	∞	∞	∞

Auch mit Poison Reverse konvergieren Distanzvektorprotokolle nur langsam!

Link State Routing

- *Verteiltes, adaptives Routing*
- Grundidee: Jeder Router kennt den Zustand des gesamten Netzes und berechnet sich daraus die Routing-Tabelle
- Beispiele: OSPF (Open Shortest Path First) und IS-IS (Intermediate System - Intermediate System)
- Algorithmus:
  - Entdecken neuer Nachbarn (mit HELLO-Paket)
  - Messung der Verzögerung bzw. Kosten zu jedem Nachbarn (ECHO-Paket misst Umlaufzeit)
  - Erstellen eines „Link-State“-Pakets mit allen gelernten Daten (Sender, Liste der Nachbarn mit Verzögerung, Zeitstempel)
  - Periodisches oder ereignisgesteuertes Fluten dieses Paketes im Netz (z.B. bei neuem Nachbar, Leitungsausfall, etc.)
  - Auf jedem Router Berechnung des kürzesten Pfades zu allen anderen Routern (mit z.B. Dijkstra-Algorithmus)

Dijkstra-Algorithmus



- Idee: Graph in Baum umwandeln
- Knoten, für den die Routing-Tabelle berechnet werden soll als Wurzel eines Baums
- Knoten mit der kleinsten Entfernung hinzufügen
- Schrittweise weiter, bis alle Knoten im Baum sind
- Entfernung zu einem Knoten = Summe entlang des Pfades im Baum

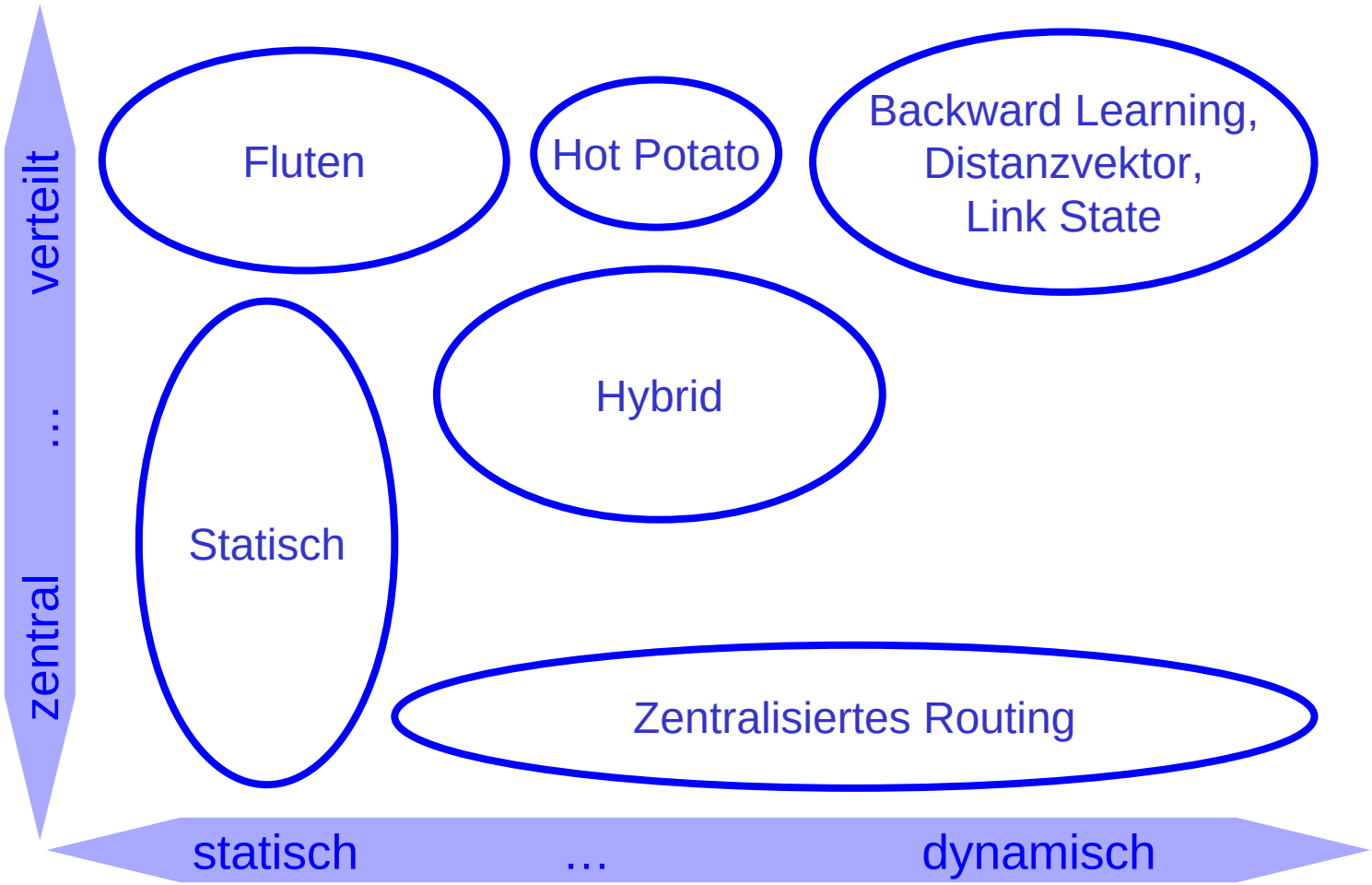
Praktische Probleme

- Oszillation der Verkehrsströme
  - Hängen die Kosten eines Links von der Verkehrslast ab, können Oszillationen entstehen
  - Lösung wäre Lastverteilung auf verschiedene Routen mit ähnlichen Kosten. (Wird kaum in der Praxis angewandt.)
- Oszillation durch Austausch der Routing-Informationen
  - Werden die Link-State-Informationen oder die Routing-Tabellen (bei Distanzvektor-Verfahren) in konstanten Zeitintervallen verteilt, synchronisiert sich das ganze Netz.
  - Leitungen periodisch durch Informationsaustausch belastet.
  - Router periodisch durch Routenberechnung überlastet.

Link-State vs. Distanz-Vektor

- Komplexität der Kontroll-Dateneinheiten
  - Bei Link-State muss jeder Knoten die Kosten aller Links kennen.
  - Bei Distanz-Vektor werden nur zusammengefasste Informationen an benachbarte Knoten weitergegeben.
- Konvergenzgeschwindigkeit
  - Link-State: Schnelle Konvergenz zu konsistentem Zustand.
  - Distanz-Vektor: Langsame Konvergenz. Count-to-Infinity-Problem!
- Robustheit
  - Link-State berechnet Routen verteilt, Fehler bleibt ggf. begrenzt bzw. leichter zu erkennen
  - Bei Distanz-Vektor-Algorithmen kann ein Knoten inkorrekte Pfade zu allen Zielen verbreiten
- Gewinner?
  - Link-State konvergiert schneller und ist robuster
  - Distanz-Vektor einfacher zu implementieren

Routingverfahren im Überblick



Questions?

Thomas Fuhrmann

CS VIII – Network Architectures
Technical University Munich, Germany

IBDS System Architecture
University of Karlsruhe, Germany

fuhrmann@net.in.tum.de