Master Course
Computer Networks
IN2097

Prof. Dr.-Ing. Georg Carle
Florian Wohlfart

Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Recap: NAT Pros and Cons

- NAT was only thought as a temporary solution

- NAT advantages:
  - ~65000 simultaneous connections with a single LAN-side address!
    → Helps against the IP shortage
  - Devices inside local net not explicitly addressable/visible by the outside world (a security plus)

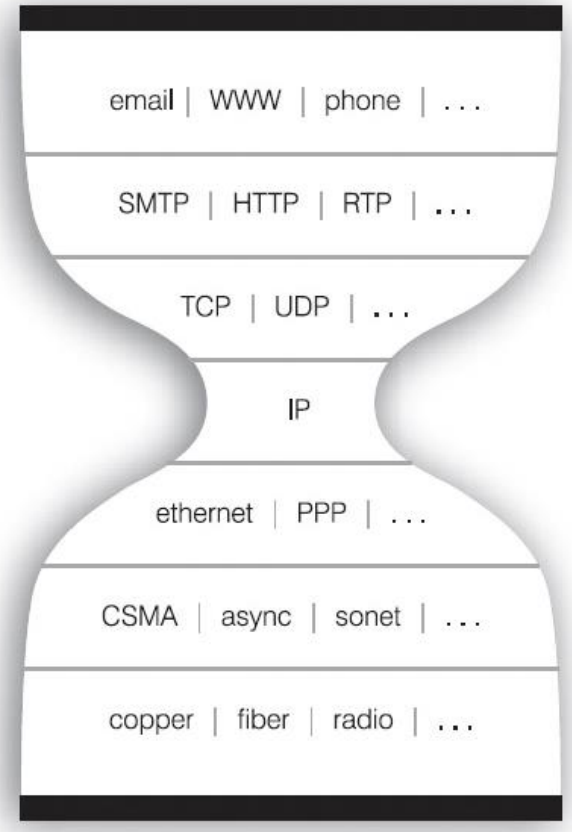
- NAT is controversial:
  - Ports should address applications, not hosts
  - Routers should only process up to layer 3
    →Violates end-to-end principle
  - Causes problems for certain applications

The IP address shortage is becoming even worse

- **More and more devices connect to the Internet**
  - PCs
  - Cell phones
  - Internet radios
  - TVs
  - Home appliances
  - Future: sensors, cars...

- With NAT, every NAT router needs an IPv4 address

- **→ ISPs still run out of global IPv4 addresses**

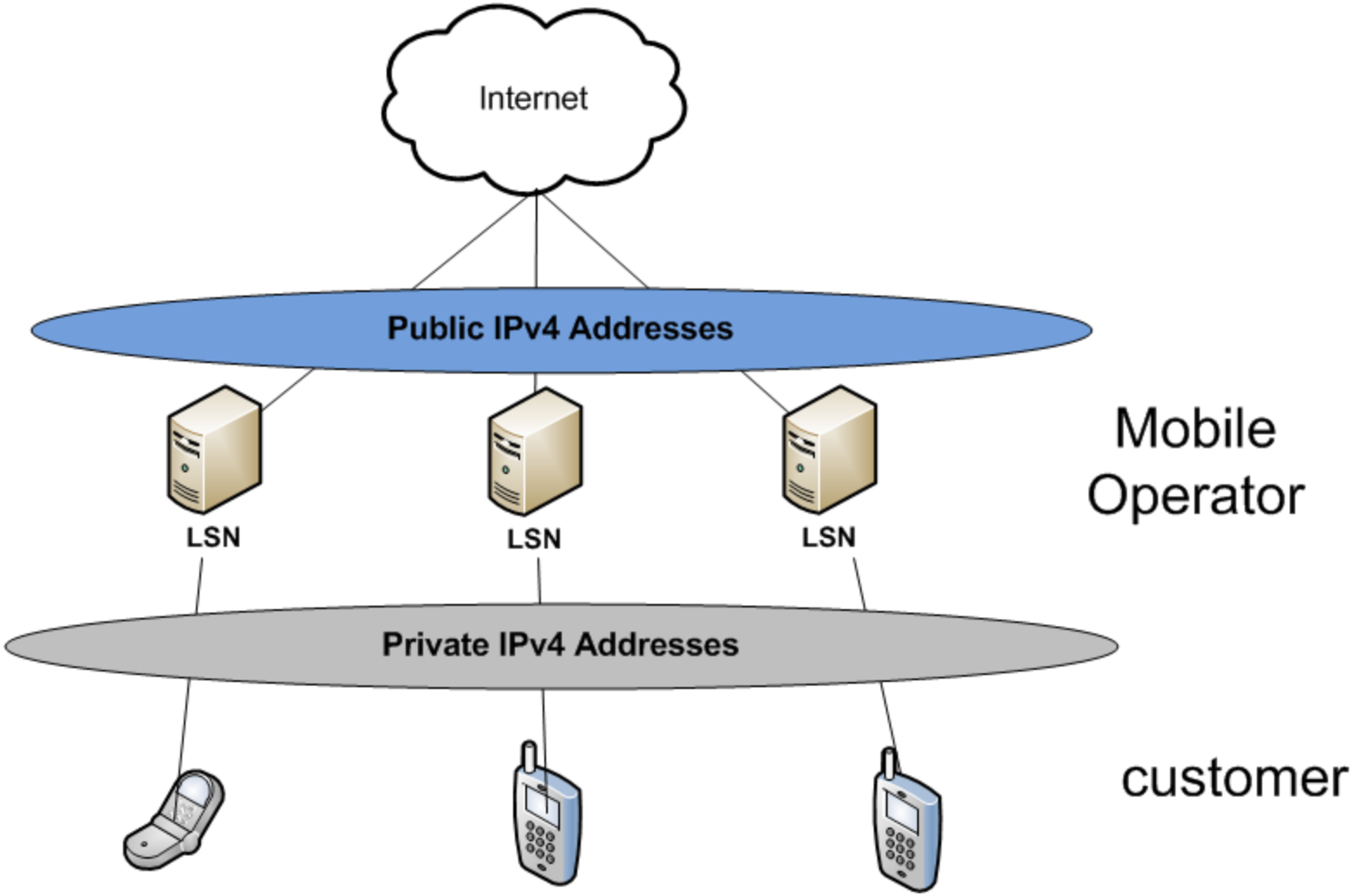


Large Scale NAT (LSN)

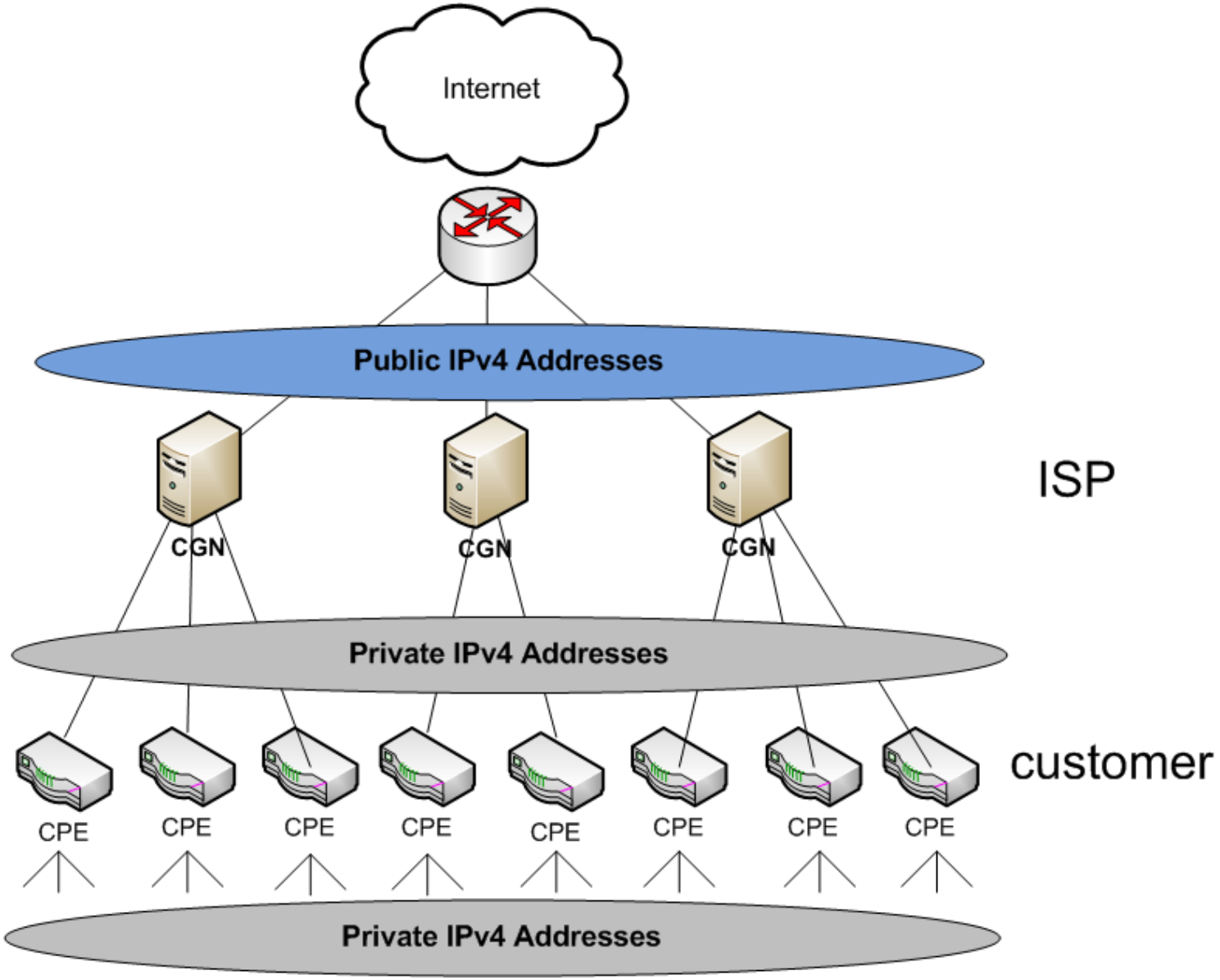
- **Facts**
  - ISPs run out of global IPv4 addresses
  - Many hosts are IPv4 only
    - 2.4% of Google users have IPv6 support [1]
  - Not all content in the web is (and will be) accessible via IPv6
    - 6.3% of the Top 1M Websites [2]
- **Challenges for ISPs**
  - Access provisioning for new customers
  - Allow customers to use their IPv4 only devices/CPEs
  - Provide access to IPv4 content
- **Approach**
  - Move public IPv4 addresses from customer to provider
  - **Large Scale NAT (LSN)** / Carrier Grade NAT (CGN) at provider for translating addresses

[1] http://www.google.com/ipv6/statistics.html
[2] http://www.employees.org/~dwing/aaaa-stats/

Large Scale NAT already common today



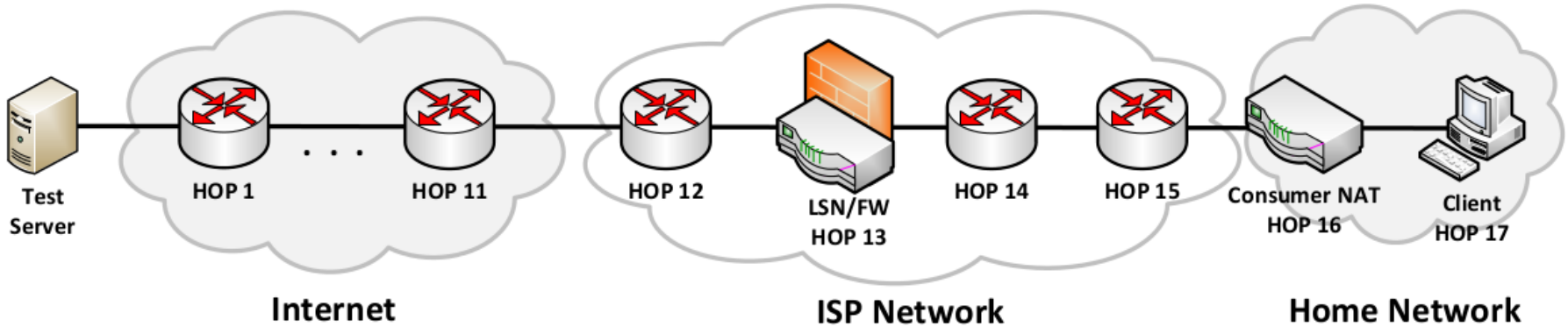
Cascaded NAT: NAT 444 (I)



Example – Vodafone LTE Network

- LTE uses the mobile network for Internet access in remote areas → often cascaded NAT

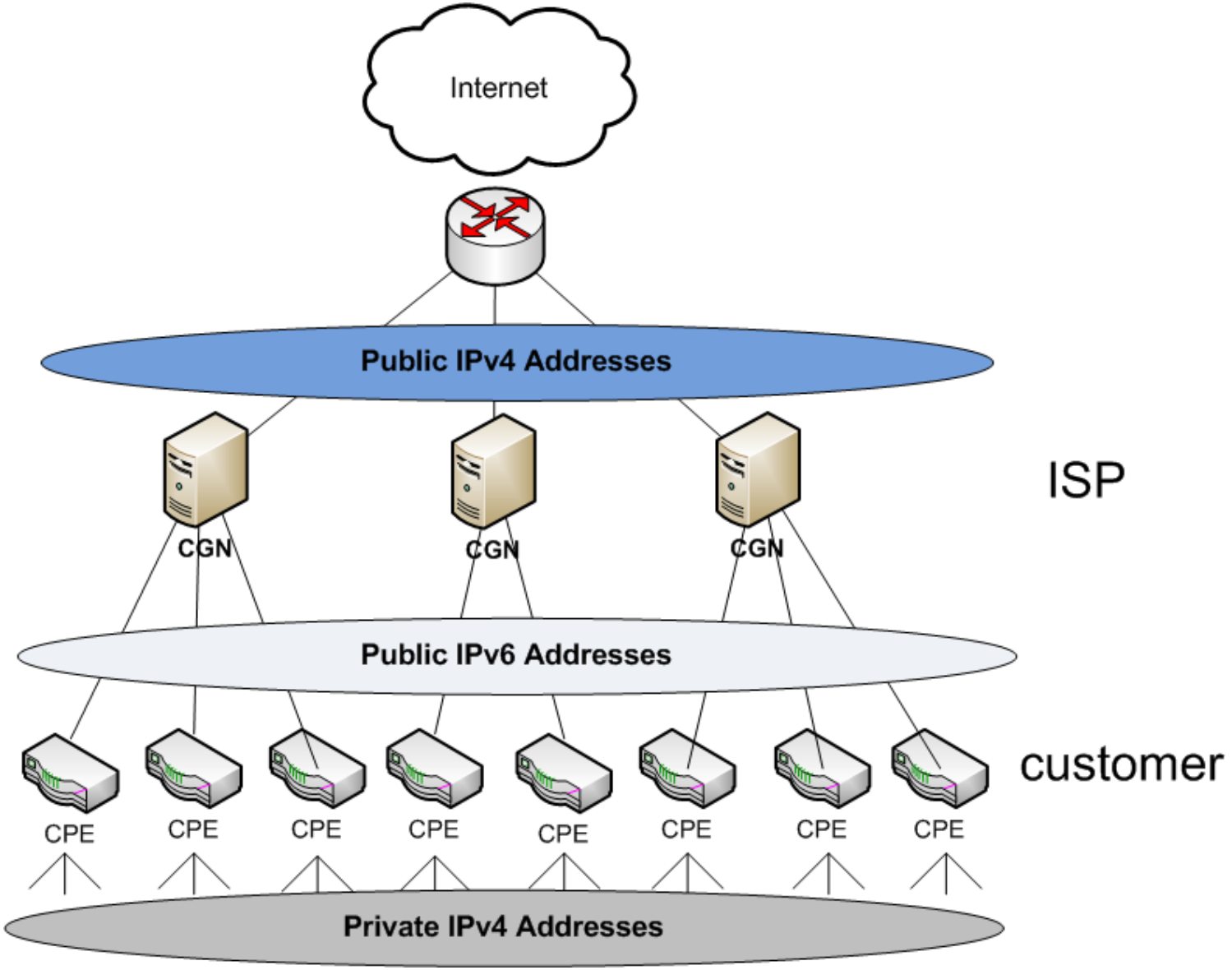
- Technique to detect network topology
  - Deployed in NAT Analyzer (nattester.net.in.tum.de)
  - Test server @TUM
  - Test client in Vodafone LTE network



Cascaded NAT: NAT 444 (II)

- Easiest way to support new customers
  - Immediately available
  - No changes at CPEs (Customer Premises Equipment)
- Problems:
  - Address overlap → same private IP address on both sides
  - Hairpinning necessary: firewalls on CPE may block incoming packets with a private source address
- Solutions
  - Declare a range of public IP addresses as „ISP shared“ and reuse it as addresses between CGN and CPE
  - NAT 464: IPv6 between CPE and CGN
    - Problem: CPEs must implement NAT64

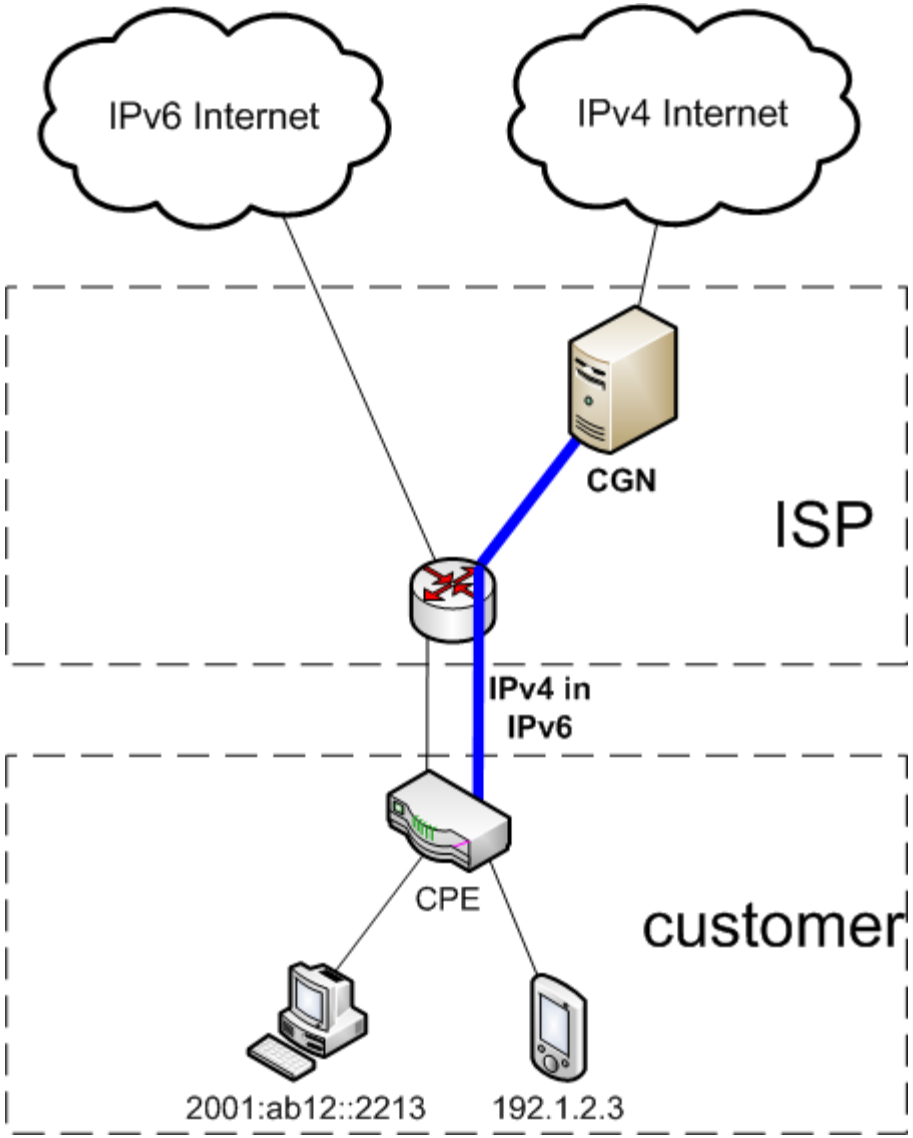
Cascaded NAT: NAT 464



Dual Stack Lite

- Mixture of NAT 444 and NAT 464
- IPv4 in IPv6 tunnel between CPE and ISP
  - No need for protocol translation in the CPE
  - No cascaded NATs
- Allows to deploy IPv6 in the ISP network while still supporting IPv4 content and IPv4 customers
  - As IPv6 devices become available they can be directly connected without the need for a tunnel
- Mainly pushed by Comcast (in IETF)
- Already deployed by providers, e.g. Unitymedia (Kabel BW)

Dual Stack Lite



Large Scale NAT – Challenges

- Mainly: how to manage resources
  - Ports (number of ports, allocation limit (time)
  - Addresses
  - Bandwidth
  - Legal issues (logging)

- NAT behavior
  - Desired: first packet reserves a bin for the customer → less logging effort
  - IP address pooling: random vs. paired (same ext IP for internal host)
    - Pairing between external and internal IP address

- Impacts of double NAT for users
  - Blacklisting as done today (based on IPs) will be a problem
  - No control of ISP NATs

- Possible Approaches
  - Small static pool of ports in control of customer
  - Needs configuration/reservation/security protocols

NAT Conclusion

- NAT helps against the shortage of IPv4 addresses
- NAT works as long as the server part is in the public internet
- P2P communication across NAT is difficult
- NAT behavior is not standardized
  - Keep that in mind when designing a protocol
- Many solutions for the NAT-Traversal problem
  - None of them works with all NATs
  - Framework can select the most appropriate technique
- New challenges with the transition to IPv6

IPv6

IPv6 – Motivation

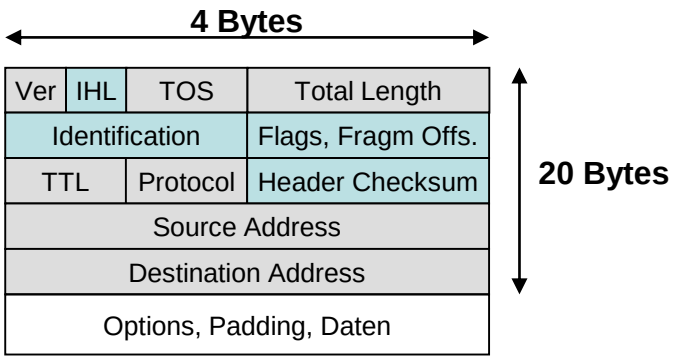
- We are running out of IPv4 addresses
  - Increased address space
- Growing routing tables
  - IPv4 address space is fragmented
  - Nearly 500.000 entries in the BGP routing table
- Simplified processing by routers
  - Simplified packet header
  - No fragmentation
  - No checksum
- Protocol redesign allows various improvements
  - Auto configuration, quality of service, …

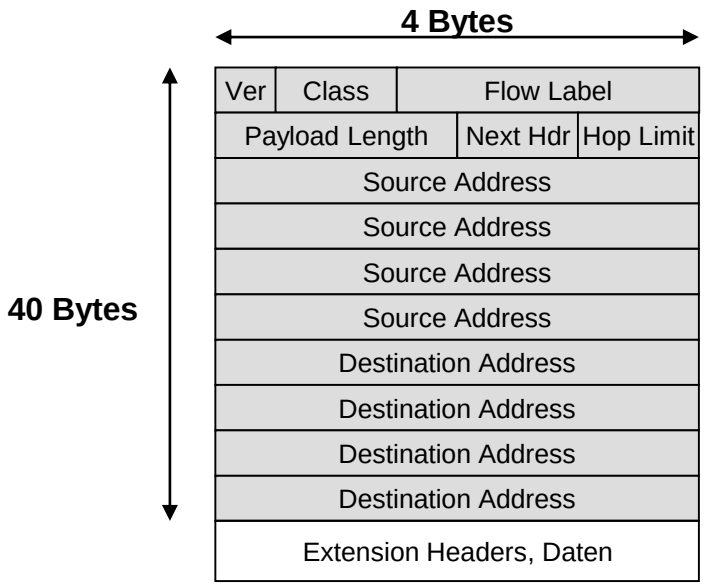
IPv6 – Overview

- Standardized in RFC 2460 „ Internet Protocol, Version 6 (IPv6)“
  - Released in December 1998
  - First version RFC 1883, from 1995

- Increased address space → more hierarchy for routing
- Simplified packet header
- Multicast instead of broadcast
- Stateless address autoconfiguration (SLAAC)
- Mobility and multihoming
- Support for Network-layer security (IPsec)
- Quality of Service support

IPv4 and IPv6 Header

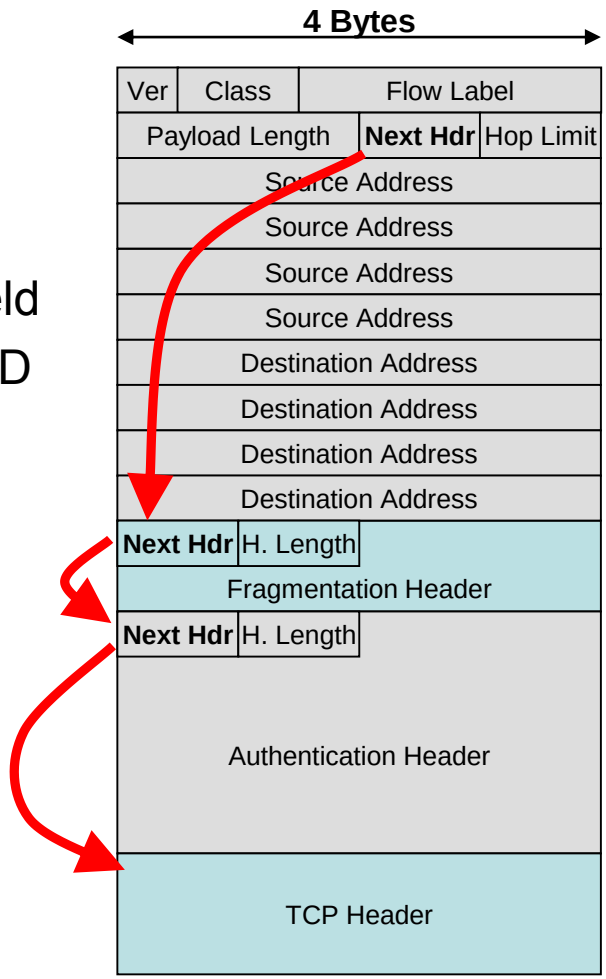




Version = 4	Version = 6
IHL (Internet Header length)	Always 40 Bytes
TOS (Type of Service)	Traffic Class + Flow Label for identifying Flows
Total Length (Header + Paket)	Payload Length (without Header)
Identification, Flags, Fragment Offset	Fragmentation only via Extension Header
TTL (Time To Live)	Hop Limit
Protocol (e.g.. TCP)	Next Header (e.g. TCP or an Extension Header)
Header Checksum	Not needed
Source / Destination each 32 Bit	Source / Destination each 128 Bit

Extension Headers

- The length of an IPv6 header is always 40 bytes
  - Options only via extension headers
  - Each IPv6 header has a „next header“ field
  - The last „next header“ field contains the ID of the layer 4 protocol (6=TCP, 17=UDP)



IPv6 Addresses Overview

- **Address field extended from 32 to 128bit**
  - 340 282 366 920 938 463 463 374 607 431 768 211 456 addresses
  - In theory: 2^{95} addresses per person

  ➔ Allow a better, systematic, hierarchical allocation of addresses and efficient route aggregation

- **Notation defined in RFC 2491**
  - Colon hexadecimal notation
    - 4 bits per char (0-F), after 4 chars blocks are separated by colon
    - Leading zeros in a group may be omitted (but at least one digit per group must be left)

      `2001:db8:85a3:8d3:1319:8a2e:370:7344`
    - Groups of zeros:
      `2001:db8:0000:0000:0000:0000:1428:57ab`

      `-> 2001:db8::1428:57ab`
  - Ports in URLs
    - `http://[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:80/`

IPv6 Unicast Addresses

- Defined in RFC3587
  - 64bit Network Prefix(es)
  - 64bit Interface Identifier

- Global Network Prefix
  - Provider assigns ***/48*** to ***/64*** subnet to customer
  - From the 2000::/3 address space → 1/8 of total address space
- Interface Identifier
  - Must be unique in the local network
  - Derived from MAC address or chosen random (privacy!)
- Loopback-Address ::1
- Link-Local Addresses (not routed)
  - Prefix: fe80::/64

IPv6 Multicast Addresses

- IPv6 uses multicast instead of broadcast
  - More targeted addressing of hosts
  - Predefined multicast addresses

- All Nodes (= broadcast)
  - FF01:0:0:0:0:0:0:1 ➔ All Nodes Host-Local
  - FF02:0:0:0:0:0:0:1 ➔ All Nodes Link-Local

- All Routers
  - FF02:0:0:0:0:0:0:2 ➔ All Routers Link-Local
  - FF05:0:0:0:0:0:0:2 ➔ All Routers Site-Local

- Continue like this:
  - FF05:0:0:0:0:0:0:101 ➔ All NTP-Servers Site-Local

Neighbor Discovery Protocol (NDP)

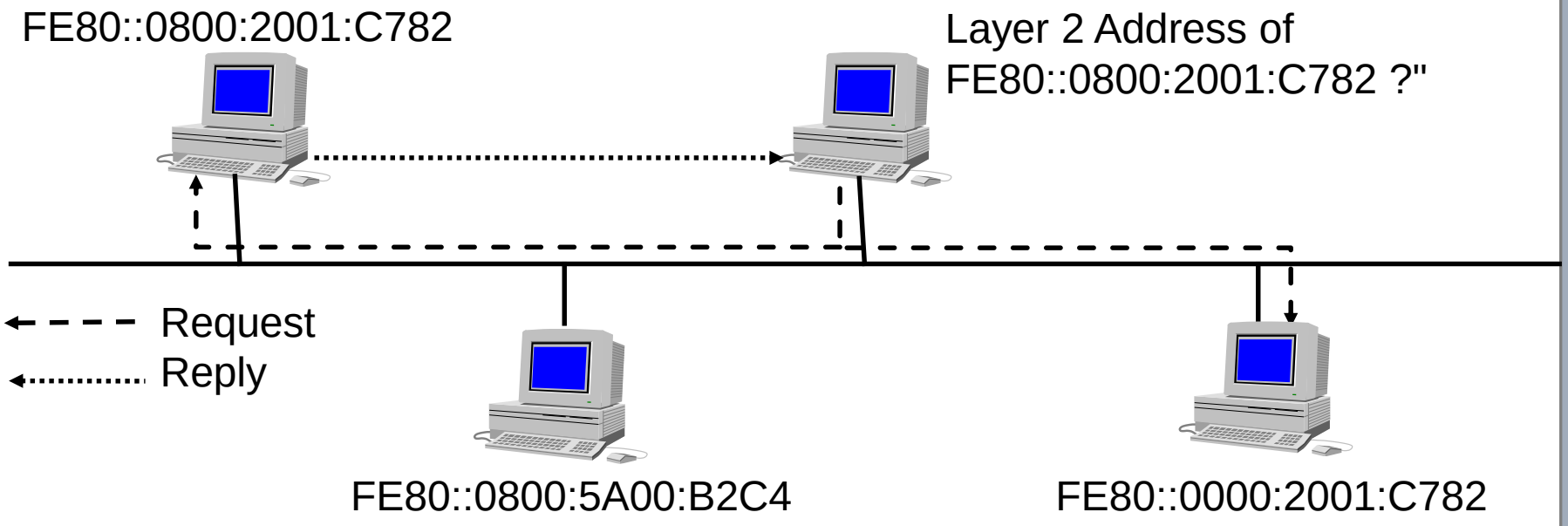
- ARP-replacement and stateless address autoconfiguration
- NDP uses different ICMPv6 messages for link local communication:

  - **Router Advertisement messages**
    - Routers advertise their presence together with various link parameters. RA contain prefixes, suggested hop limit values…
  - **Router Solicitation messages**
    - Request for a Router Advertisement message
    - Used when a link becomes enabled
  - **Neighbor Solicitation messages**
    - Sent by a node to determine the link-layer address of a neighbor (ARP replacement)
    - Also used for duplicate address detection
  - **Neighbor advertisement messages**
    - Response to a Neighbor Solicitation message

Stateless Address Autoconfiguration (SLAAC)

1. Interface is activated and generates a Link local address
   - FE80::/64 prefix + Interface Identifier
2. Node sends out a Neighbor Solicitation Message to check if its link local address is unique
   - Destination: its own link local address
3. If address is unique the node is able to communicate with its neighbors, if not manual configuration is needed
4. Wait for a router advertisement (or send out a router solicitation message if you can‘t wait)
   - Use announced prefix to generate a globally routeable address

Neighbor Solicitation



- **Problem:**
  - Resolve IPv6 address to Layer 2 address (replaces ARP)
- **Solution:** *request via Neighbor Solicitation*
  - All nodes listen on *Solicited Nodes Address* (Prefix `FF02::1:FF` + last 24 Bit (e.g. `01:C782`) of their Unicast-Address, e.g. `FF02::1:FF01:C782`)
  - Send request to *Solicited Nodes Address* of the desired destination (L2-Multicast)
  - Destination node replies with its Layer 2 address

Transition to IPv6

- New Layer 3 Protocol requires changes for other protocols
  - New socket API that supports IPv6
  - Routing protocols must support IPv6
  - IPv6 DNS (AAAA record)

- Slow Transition, many strategies possible
  - Dual Stack
    - Each node implements both stacks and decides on every incoming packet which one should handle it
  - IPv6 Tunneling
    - Tunnel IPv6 in IPv4 packets if we have to cross an IPv4 network
  - Dual Stack Lite

Internet Routing

Short note on pronunciation of the word “routing”

- [‘ruːtɪŋ] /r-oo-ting/ = British English
- [‘raʊdɪŋ] /r-ow-ding/ = American English
- Both are correct!

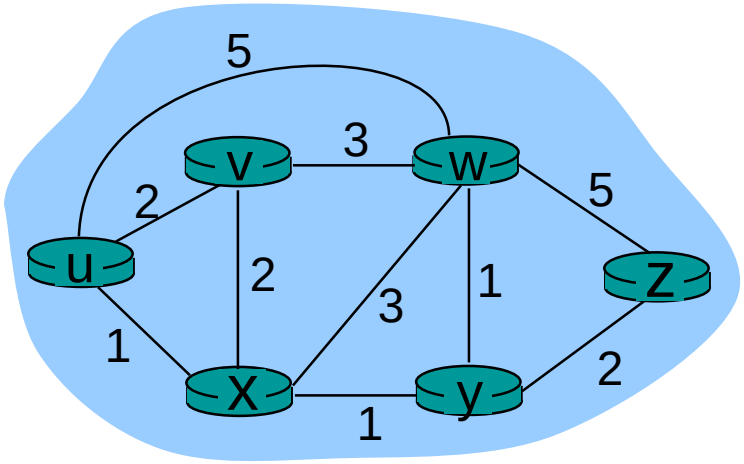
Routing - Topics

- Routing and forwarding
- Routing algorithms recapitulated
  - Link state
  - Distance Vector
  - Path Vector
- Intradomain routing protocols
  - RIP
  - OSPF
- Interdomain routing
  - Hierarchical routing
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering
- Broadcast and Multicast routing

Routing ≠ Forwarding

- Routing:
  - Process of determining the best path for a specific type of packets (usually: all packets with the same destination) through the network
  - Performed jointly by routers of a network by exchanging messages
  - Analogy: Read street map, plan journey
- Forwarding:
  - Process where a router relays a packet to a neighbouring router. Selection of the neighbouring router depends on the previous routing protocol calculations
  - Performed by one router on one packet
  - Analogy: Read a street sign and determine whether to turn right
- In practice, this distinction in terminology is often ignored
  - “If router A routes packet X, then …”
  - Actually, it doesn‘t – it *forwards* X.

Graph abstraction



Graph: G = (N,E)

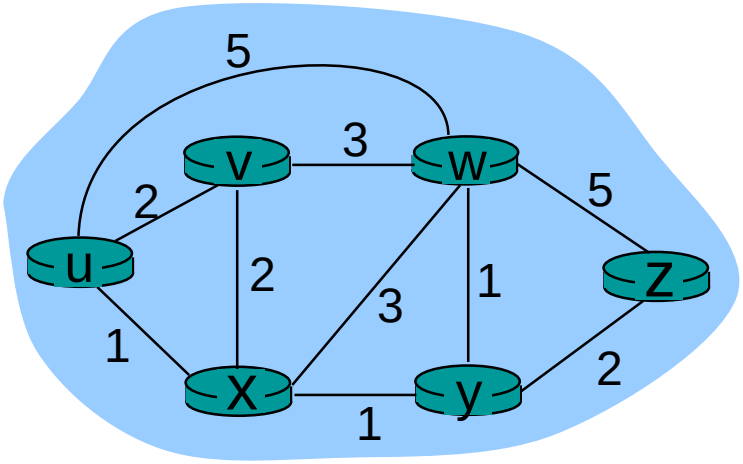
N = nodes = set of routers = { u, v, w, x, y, z }

E = edges = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x') =: cost of link (x,x')
  e.g.: c(w,z) = 5

- cost could always be 1,
- or inversely related to bandwidth,
- or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

Global or decentralized information?

Global:

- All routers have complete topology and link cost info
- *link state* algorithms (L-S)

Decentralized:

- Router only knows physically-connected neighbors and link costs to neighbors
- Iterative process of computation = exchange of info with neighbours
- *distance vector* algorithms (D-V)
- Variant: *path vector* algorithms

Static or dynamic?

Static:

- Routes change slowly over time

Dynamic:

- Routes change more quickly
  - periodic update
  - in response to link cost changes

A broader routing classification

- Type of algorithm: Link State, Distance Vector, Path Vector, …
- Scope:
  - Intradomain
  - Interdomain
  - Special purpose (e.g., sensor network)
- Type of traffic: Unicast vs. multicast
- Type of reaction: “Static” vs. Dynamic/adaptive
  - Warning: “Dynamic routing” is a fuzzy term:
    a) Dynamic ≔ reacts to topology changes (state of the art)
    b) Dynamic ≔ reacts to traffic changes (possibly even better, but most protocols don’t do that!)
- Trigger type:
  - Permanent routing (standard)
  - On-demand routing: only start routing algorithm if there is traffic to be forwarded (e.g., some wireless ad-hoc networks)

A Link-State Routing Algorithm

- Network topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same information (…after all information has been exchanged)
- Each node independently computes least-cost paths from one node (“source”) to all other nodes
  - Usually done using Dijkstra’s shortest-path algorithm
    - refer to any algorithms & data structures lecture/textbook
    - n nodes in network $\Rightarrow$ $O(n^2)$ or $O(n \log n)$
  - Gives forwarding table for that node
- Result:
  - All nodes have the same information,
  - … thus calculate the same shortest paths,
  - … hence obtain consistent forwarding tables

Distance Vector Algorithm

- No node knows entire topology
- Nodes only communicate with neighbours (i.e., no broadcasts)
- Nodes *jointly* calculate shortest paths
  - Iterative process
  - Algorithm == protocol
- Distributed application of Bellman-Ford algorithm
  - c.f. algorithms&data structures lecture/textbook

Bellman-Ford Equation (dynamic programming)

Let

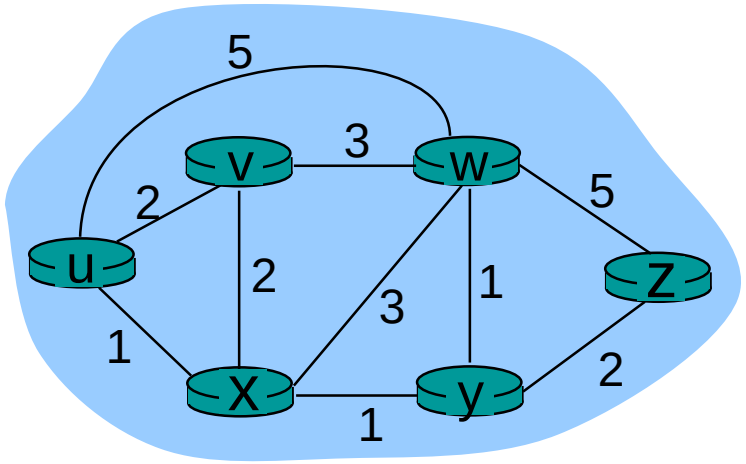
- $c(x,y) :=$ cost of edge from x to y
- $d_x(y) :=$ cost of least-cost path from x to y
- Set to ∞ if no path / no edge available

Then

$$d_x(y) = \min \{c(x,v) + d_v(y)\}$$

where min is taken over all neighbours v of x

Bellman-Ford example



We can see that
$d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z),$$
$$c(u,x) + d_x(z),$$
$$c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5,$$
$$1 + 3,$$
$$5 + 3\} = 4$$

Node that calculated minimum is next hop in shortest path
→ forwarding table

Distance Vector Algorithm

- Define $D_x(y)$:= estimate of least cost from x to y
- Node x knows cost to each neighbour v: $c(x,v)$
- Node x maintains distance vector $\overrightarrow{D_x} := [\, D_x(y) : y \in N \,]$
  (N := set of nodes)
- Node x also maintains copies of its neighbours' distance vectors
  - Received via update messages from neighbours
  - For each neighbour v,
    x knows $\overrightarrow{D_v} = [\, D_v(y) : y \in N \,]$

Distance vector algorithm

Basic idea:

- From time to time, each node sends its own distance vector estimate D to its neighbours
  - Asynchronously
- When a node x receives new DV estimate from neighbour, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under minor, natural conditions, these estimates $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm

Iterative, asynchronous:

Each local iteration caused by:

- Local link cost change
- DV update message from neighbour

Distributed:

- Each node notifies neighbours *only* when its DV changes
  - neighbours then notify their neighbours if this caused *their* DV to change
  - etc.

Usually some waiting delay between consecutive updates

Each node:

Forever:

wait for (change in local link cost *or* message arriving from neighbour)

recompute estimates

if (DV to any destination has changed) { *notify* neighbours }

Distance Vector Algorithm

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

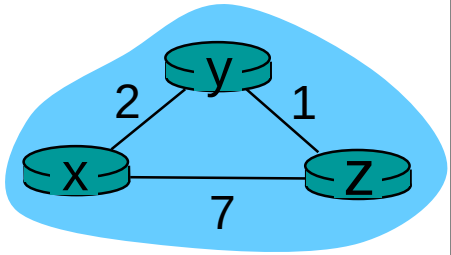
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞



node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

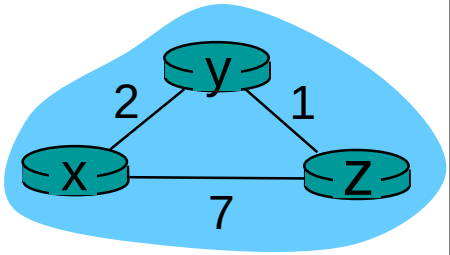
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

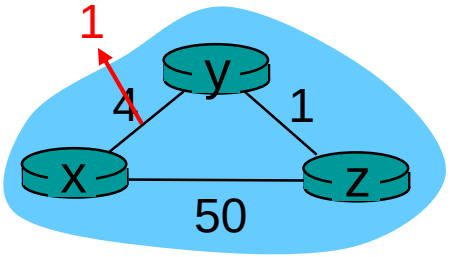


time

Distance Vector: link cost changes (1)

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbours



"good news travels fast"

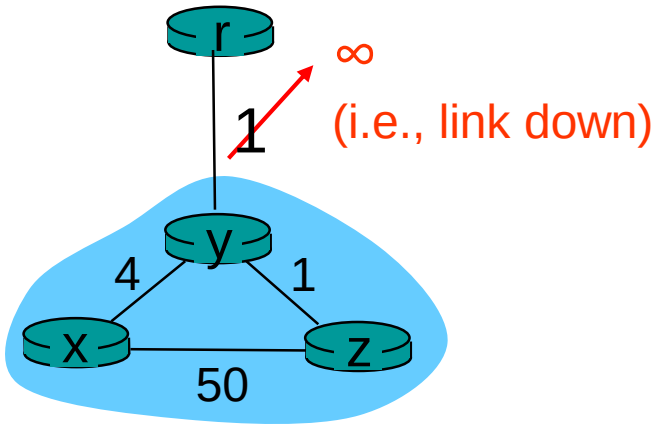
At time t_0, y detects the link-cost change, updates its DV, and informs its neighbours.

At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbours its new DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector: link cost changes (2)

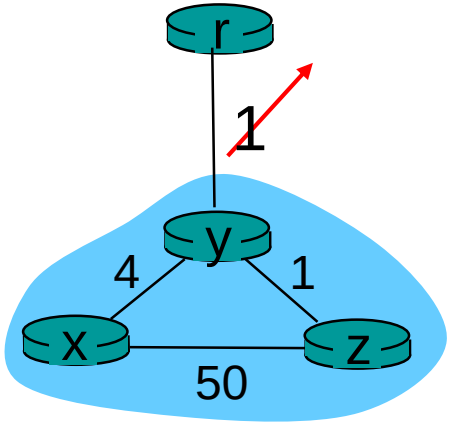
- But: bad news travels slow
- In example: Many iterations before algorithm stabilizes!
  1. Cost increase for *y*→*r*:
     - *y* consults DV,
     - *y* selects "cheaper" route via *z* (cost 2+1 = 3),
     - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)



  2. *z* detects cost increase for path to *r*:
     - was 1+1, is now 3+1
     - Sends update to *y* and *x* (cost to *r* now 4 instead of 2)
  3. *y* detects cost increase, sends update to *z*
  4. *z* detects cost increase, sends update to *y*
  5. ….
- Symptom: "count to infinity" problem

Distance Vector: Problem Solutions…

- Finite infinity: Define some number to be ∞ (in RIP: ∞ := 16)
- Split Horizon:
  - Tell to any neighbour that is part of a best path to a destination that the destination cannot be reached
  - If *z* routes through *y* to get to *r*
    z tells *y* that its own distance to *r* is infinite
    (so *y* won't route to *r* via *z*)
- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned



…that only half work

- Mechanisms can be combined
- Both mechanisms can significantly increase number of routing messages
- Often help, but cannot solve all problem instances
  - Think yourselves: Come up with a topology where this does not help
  - Try it – it's not hard and a good exercise

Comparison of LS and DV algorithms

Message complexity

- LS: with *n* nodes, *E* links, O(*nE*) messages sent to all nodes
- DV: exchange between neighbours only

Speed of Convergence

- LS: O(n^2) algorithm requires O(*nE*) messages
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagates through network

Path Vector protocols

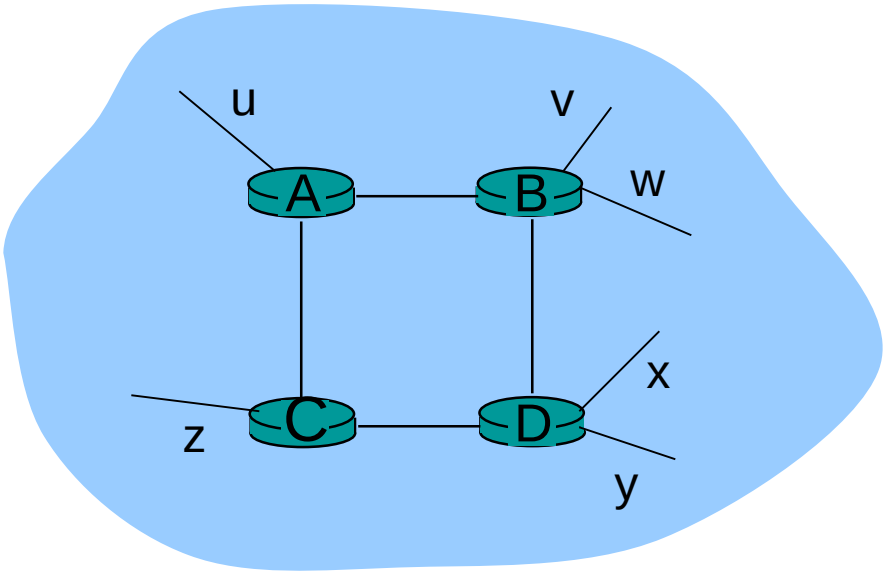
- Problem with D-V protocol:
  Path cost is “anonymous” single number; does not contain any topology information
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
    - E.g., count number of hops on the path
  - Easy loop detection: Does my node ID already appear in the path?
- Not used very often
  - only in BGP …
  - … and BGP is much more complex than just paths

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol — DV (typically small systems)
  - OSPF: Open Shortest Path First — hierarchical LS (typically medium to large systems)
  - IS-IS: Intermediate System to Intermediate System — hierarchical LS (typically medium-sized ASes)
  - (E)IGRP: (Enhanced) Interior Gateway Routing Protocol (Cisco proprietary) — hybrid of LS and DV

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops, ∞ ≔ 16)
- Used in small networks only



From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

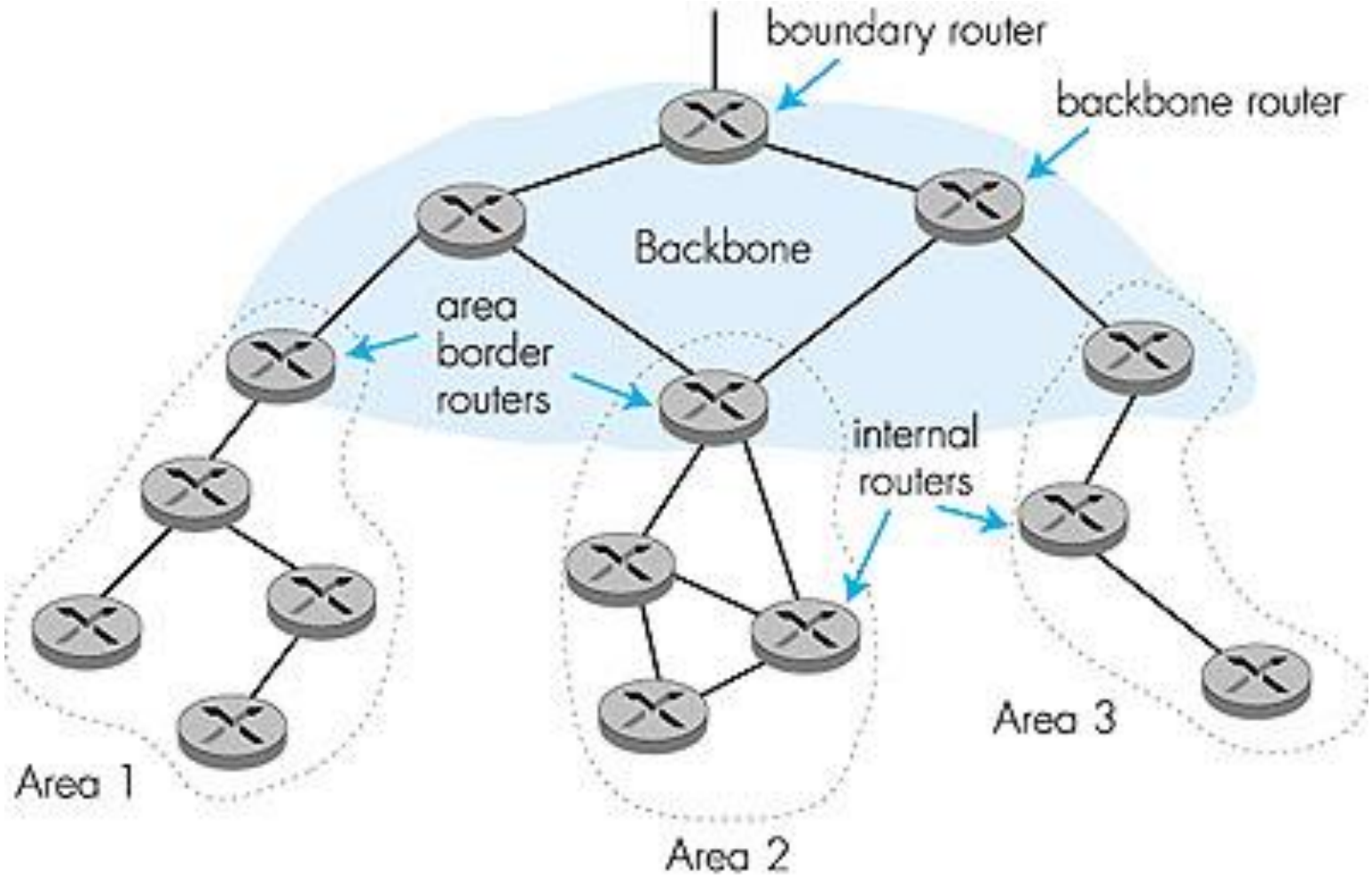
OSPF (Open Shortest Path First)

- "Open": publicly available
  (vs. vendor-specific, e.g., EIGRP = Cisco-proprietary)
- Uses Link State algorithm
  - LS packet dissemination (broadcasts)
  - Unidirectional edges (⇒costs may differ by direction)
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbour router
- Advertisements disseminated to entire AS (via flooding)
  - (exception: hierarchical OSPF, see next slides)
  - carried in OSPF messages directly over IP
    (instead of using TCP or UDP)

OSPF “advanced” features (not in, e.g., RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP): *ECMP* (equal-cost multipath)
- For each link, multiple cost metrics for different Type of Service (TOS):
  e.g., satellite link cost set to “low” for best effort, but to “high” for real-time traffic like (telephony)
- Integrated unicast *and* multicast support:
  - Multicast OSPF (MOSPF)
  - Uses same topology data base as OSPF ➔ less routing protocol traffic
- Hierarchical OSPF in large domains
  - Drastically reduces number of broadcast messages

Hierarchical OSPF



Hierarchical OSPF

- OSPF *can* create a two-level hierarchy
  - (similar, but not identical to to inter-AS and intra-AS routing within an AS)
- Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology;
    but only knows coarse direction to networks in other areas (shortest path to border router)
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border and Boundary routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other Ases
  - "The outside world" ≈ another area