CS118 Discussion 1D, Week 7

Zengwen Yuan
Boelter Hall 2760, Friday 4:00—5:50 p.m.
Outline

- Network data plane (cont’d)
  - Switching, IPv4/IPv6, DHCP, NAT
- Network control plane
  - Routing
    - Link state routing (OSPF)
    - Distance vector routing (RIP)
    - BGP
    - ICMP
- Midterm
DHCP: Dynamic Host Configuration Protocol

- Dynamically allocates the following info to a host
  - IP address on subnet for the host
  - IP address for default router ("first-hop" router)
  - Subnet mask
  - IP address and name for DNS caching resolver
- Allows address reuse
DHCP: operations

- Host broadcasts “DHCP discovery” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- Host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

Important example on Chapter 4 slides 45—46!
NAT (network address translation)

- Depletion of IPv4 addresses — short-term solution
  - IP tunneling?
- Use private IP addresses
- Side-benefit: security
- How to achieve?
  - <public IP:port> — <private IP:port> mapping
NAT: detail

- outgoing packets:
  - replace (source IP address, source port #) of every outgoing packet to (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
  - remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

- incoming packets:
  - replace (destination NAT IP address, destination port #) of every incoming packet with corresponding (source IP address, port #) stored in NAT table
NAT: downside

- Increased complexity
- Single point of failure
- Cannot run services inside a NAT box
IPv6

IPv6 Header Format (RFC 2460)
IPv6/IPv4 differences

- Fixed-length 40 byte header
  - length field excludes header
  - Header Length field eliminated
- Address length: 128 bits
- Priority: usage yet to be finalized
- Flow Label: identify packets in same flow
- Next header: identify upper layer protocol for data
- Options: outside of the basic header, indicated by Next Header field
- Header Checksum: removed
IPv6 address format (optional)


- Can skip leading zeros of each word: 2607:F010:3f9:0:0:0:4:1

- Can skip one sequence of zero words (compressed representation), e.g., 2607:f010:3f9::4:1

- Can leave the last 32 bits in dot-decimal: 2607:f010:3f9::0.4.0.1

- Can specify a prefix by /length: 2607:f010:3f9::/64
Special IPv6 addresses (optional)

- ::/128 - Unspecified
- ::1/128 - Loopback
- ::ffff:0:0/96 - IP4-mapped address
- 2002::/16 - 6to4
- ff00::/8 - Multicast
- fe80::/10 - Link-Local Unicast
Routing: concepts

• Global or decentralized information?
  • global: all routers have complete topology, link cost info
  • algorithm?
Routing: concepts

• Global or decentralized information?

  • global: all routers have complete topology, link cost info

  • “link state” algorithms
Routing: concepts

- Global or decentralized information?
  - global: all routers have complete topology, link cost info
  - “link state” algorithms
  - decentralized: router knows physically-connected neighbors, link costs to neighbors; iterative process of computation, exchange of info with neighbors
  - algorithm?
Routing: concepts

• Global or decentralized information?
  • global: all routers have complete topology, link cost info
    • “link state” algorithms
  • decentralized: router knows physically-connected neighbors, link costs to neighbors; iterative process of computation, exchange of info with neighbors
    • “distance vector” algorithms
Link state routing

• Dijkstra’s algorithm

  • net topology, link costs known to all nodes

  • computes least cost paths from one node (‘source”) to all other nodes

  • iterative: after k iterations, know least cost path to k destinations
Initialization:

1. \( N' = \{u\} \)
2. for all nodes \( v \)
3.     if \( v \) adjacent to \( u \)
4.         then \( D(v) = c(u,v) \)
5.     else \( D(v) = \infty \)
6. Loop
7.     find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
8.     add \( w \) to \( N' \)
9.     update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \):
10. [Link cost update heuristic from Dijkstra algo.]
11. until all nodes in \( N' \)

\( c(x, y) \): link cost from node \( x \) to \( y \); \( c(x, y) = \infty \) if not direct neighbors
\( D(v) \): current value of cost of path from source to destination \( v \)
\( p(v) \): predecessor node along path from source to \( v \)
\( N' \): set of nodes whose least cost path definitively known
Link state routing: algorithm

1 Initialization:
2 \[ N' = \{u\} \]
3 for all nodes \( v \)
4     if \( v \) adjacent to \( u \)
5         then \( D(v) = c(u,v) \)
6     else \( D(v) = \infty \)
7
8 Loop
9     find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
10    add \( w \) to \( N' \)
11    update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \):
12 \[ D(v) = \min( D(v), D(w) + c(w,v) ) \]
13 until all nodes in \( N' \)

c(x, y): link cost from node \( x \) to \( y \); \( c(x, y) = \infty \) if not direct neighbors
D(v): current value of cost of path from source to destination \( v \)
p(v): predecessor node along path from source to \( v \)
N': set of nodes whose least cost path definitively known
Link state routing: example

• Using link state routing to setup a forwarding table for node u
Let’s work it out

<table>
<thead>
<tr>
<th>N’</th>
<th>D(v), p(v)</th>
<th>D(w), p(w)</th>
<th>D(x), p(x)</th>
<th>D(y), p(y)</th>
<th>D(z), p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>2, u</td>
<td>5, u</td>
<td>1, u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>ux</td>
<td>2, u</td>
<td>4, x</td>
<td></td>
<td>2, x</td>
<td>∞</td>
</tr>
<tr>
<td>uxy</td>
<td>2, u</td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyv</td>
<td></td>
<td>3, y</td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, y</td>
</tr>
<tr>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Let's work it out

<table>
<thead>
<tr>
<th>$N'$</th>
<th>$D(v), p(v)$</th>
<th>$D(w), p(w)$</th>
<th>$D(x), p(x)$</th>
<th>$D(y), p(y)$</th>
<th>$D(z), p(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>2, $u$</td>
<td>5, $u$</td>
<td>1, $u$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$ux$</td>
<td>2, $u$</td>
<td>4, $x$</td>
<td></td>
<td>2, $x$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$uxy$</td>
<td></td>
<td></td>
<td>3, $y$</td>
<td></td>
<td>4, $y$</td>
</tr>
<tr>
<td>$uxyv$</td>
<td></td>
<td></td>
<td>3, $y$</td>
<td></td>
<td>4, $y$</td>
</tr>
<tr>
<td>$uxyvw$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, $y$</td>
</tr>
<tr>
<td>$uxyvwz$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Distance vector routing

• Bellman-Ford equation (dynamic programming)

• let

• \( dx(y) := \text{cost of least-cost path from } x \text{ to } y \)

• then

• \( dx(y) = ? \)
Distance vector routing

• Bellman-Ford equation (dynamic programming)

• let

• \( dx(y) := \text{cost of least-cost path from } x \text{ to } y \)

• then

• \( dx(y) = \min_v \{ c(x,v) + dv(y) \} \)
Distance vector routing: example

- What’s the cost of least-cost path for $u \rightarrow z$?
Let’s work it out

• clearly:

• $dv(z) = \ ?$, $dx(z) = \ ?$, $dw(z) = \ ?$
Let’s work it out

• clearly:
  • $dv(z) = 5$, $dx(z) = 3$, $dw(z) = 3$

• According to B-F equation:
  • $du(z) = \min \{ ? \}$
Let’s work it out

• clearly:
  • $dv(z) = 5$, $dx(z) = 3$, $dw(z) = 3$

• According to B-F equation:
  • $du(z) = \min \{ c(u,v) + dv(z), c(u,x) + dx(z), c(u,w) + dw(z) \}$
Let’s work it out

• clearly:
  • \( dv(z) = 5, \ dx(z) = 3, \ dw(z) = 3 \)

• According to B-F equation:
  • \( du(z) = \min \{ c(u,v) + dv(z), c(u,x) + dx(z), c(u,w) + dw(z) \} \)
    • \( \quad = \min \{ 2 + 5, 1 + 3, 5 + 3 \} = 4 \)
Distance vector routing: key idea

- from time-to-time, each node sends its own distance vector estimate to neighbors

- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation.
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
Distance vector routing: caveat

- Count-to-infinity problem.

- Can you work out an example?

- Can you propose a solution?
  - basic idea?
Distance vector routing: split horizon

• Previous solution idea:
  
  • split horizon
    
    • if B reaches D through C, B should not tell C that B can reach D
    
    • Then C will not attempt to go through B to reach D
    
    • Are we good?
Distance vector routing: split horizon

• Previous solution idea:
  • split horizon
    • if B reaches D through C, B should not tell C that B can reach D
    • Then C will not attempt to go through B to reach D
  • Are we good?
Distance vector routing: poison reverse

- Split horizon + poison reverse

  - if A reaches D through C:
    - A tells C that A’s distance to D is infinite
    - Then C will not attempt to go through A to reach D
  - In practice, infinite == 16 hops
# Link State v.s. Distance Vector

<table>
<thead>
<tr>
<th></th>
<th>Link state</th>
<th>Distance vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>message complexity</td>
<td>with ( n ) nodes, ( E ) links, ( O(nE) ) msgs sent</td>
<td>exchange between neighbors only (convergence time varies)</td>
</tr>
<tr>
<td>convergence speed</td>
<td>( O(n^2) ) algorithm requires ( O(nE) ) msgs</td>
<td>convergence time varies (may be routing loops)</td>
</tr>
<tr>
<td>robustness</td>
<td>node can advertise incorrect link cost; each node computes only its own table</td>
<td>DV node can advertise incorrect path cost; error propagate thru network</td>
</tr>
<tr>
<td>implementation</td>
<td>OSPF</td>
<td>RIP</td>
</tr>
</tbody>
</table>
Summary

- Link-state routing (Dijkstra) algorithm:
  - each node computes the shortest paths to all the other nodes based on the complete topology map

- Distance Vector (Bellman-Ford) routing algorithm:
  - each node computes the shortest paths to all the other nodes based on its neighbors distance to all destinations
Inter-domain routing

• aggregate routers into regions

• AS: autonomous systems

• routers in same AS run same routing protocol

• “intra-AS” routing protocol

• routers in different AS can run different intra-AS routing protocol
BGP (Border Gateway Protocol)

- An inter-domain routing protocol; allows subnet to advertise its existence to rest of Internet: “I am here”

- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.

- How BGP works with intra-domain routing (e.g. OSPF)

Important BGP path advertisement example on Chapter 5 slides 49—50
BGP: routing policy

- A, B, C are provider networks
- X, W, Y are customer (of provider networks)
- X is attached to two networks.
  - It does not want to route from B via X to C
  - … so X will not advertise to B a route to C
BGP: routing policy

- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
BGP: routing policy

- A advertises path AW to B
- B advertises path BAW to X

Should B advertise path BAW to C?

- No! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
- B wants to force C to route to w via A
- B wants to route only to/from its customers!
BGP: practice problems

• Explain how loops in paths can be detected in BGP.

• BGP advertisements contain complete paths showing the AS’s the path passes through, and so a router can easily identify a loop because an AS will appear two or more times.
BGP: practice problems

• Suppose that there is another stub network V that is a customer of ISP A. Suppose that B and C have a peering relationship, and A is a customer of both B and C. Suppose that A would like to have the traffic destined to W to come from B only, and the traffic destined to V from either B or C. How should A advertise its routes to B and C? What AS routes does C receive?

• A should advertise to B two routes: A-W and A-V

• A should advertise to C only one route: A-V

Routing: summary

- Intra-domain routing V.S. inter-domain routing
  - Performance V.S. policy
- Scalability: hierarchical routing
- Distance-vector routing V.S. link-state routing
  - Fully-distributed algorithm V.S. decentralized algorithm
- Unicast V.S. multicast
SDN: software defined networking

- A logically centralized control plane
  - easier network management
  - programmable forwarding table (OpenFlow API)
  - open (non-proprietary) implementation of control plane

- Components
  - data plane switches
  - SDN controller
  - network-control apps

Hot Internet research topic;
P4 programmable switch
ICMP: Internet Control Message Protocol

- Used for feedback, status checking, error reporting at IP layer
- ICMP msgs are carried in IP packets
- **ping**: echo request/reply
- **traceroute**: nth packet has TTL = n
$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max, 52 byte packets
1 172.30.40.3 (172.30.40.3) 4.055 ms 3.017 ms 3.871 ms
2 wifi-131-179-60-1.host.ucla.edu (131.179.60.1) 2.545 ms 2.288 ms 2.714 ms
3 ra00f1.anderson--cr00f2.csb1.ucla.net (169.232.8.12) 3.653 ms 3.506 ms 3.724 ms
4 cr00f2.csb1--bd11f1.anderson.ucla.net (169.232.4.5) 3.959 ms 4.383 ms 3.483 ms
5 lax-agg6--ucla-10g.cenic.net (137.164.24.134) 3.951 ms 5.480 ms 3.840 ms
6 74.125.49.165 (74.125.49.165) 6.558 ms 3.882 ms 3.890 ms
7 108.170.247.129 (108.170.247.129) 3.192 ms
108.170.247.193 (108.170.247.193) 93.964 ms
108.170.247.161 (108.170.247.161) 3.297 ms
8 108.177.3.127 (108.177.3.127) 3.657 ms
209.85.255.73 (209.85.255.73) 3.571 ms
108.177.3.129 (108.177.3.129) 3.261 ms
9 google-public-dns-a.google.com (8.8.8.8) 5.315 ms 3.770 ms 12.165 ms
Traceroute: example

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>2360</td>
<td>16:17:48.237368</td>
<td>172.30.42.132</td>
<td>8.8.8.8</td>
<td>UDP</td>
<td>66</td>
<td>32435 Len=24</td>
</tr>
<tr>
<td>2361</td>
<td>16:17:48.541825</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2364</td>
<td>16:17:48.245989</td>
<td>172.30.42.132</td>
<td>8.8.8.8</td>
<td>UDP</td>
<td>66</td>
<td>42995 - 33436 Len=24</td>
</tr>
<tr>
<td>2365</td>
<td>16:17:48.540795</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2366</td>
<td>16:17:48.540795</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2367</td>
<td>16:17:48.542767</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2368</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2369</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2372</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2373</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2374</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2375</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2376</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2377</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2378</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2379</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2380</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2381</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2382</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2383</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2384</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2385</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2386</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2387</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2388</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2389</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2390</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2391</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2392</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2393</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2394</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2395</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2396</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2397</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2398</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2399</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
<tr>
<td>2400</td>
<td>16:17:48.542839</td>
<td>172.30.48.3</td>
<td>8.8.8.8</td>
<td>ICMP</td>
<td>70</td>
<td>Time-to-live exceeded (Time to live exceeded in transit)</td>
</tr>
</tbody>
</table>

Fragment offset: 0
Time to live: 5
Protocol: UDP [17]
Header checksum: 0x2707 [validation disabled]
Source: 172.30.42.132
Destination: 8.8.8.8