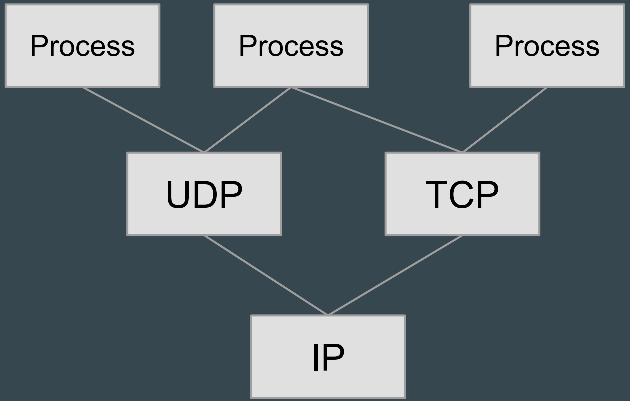
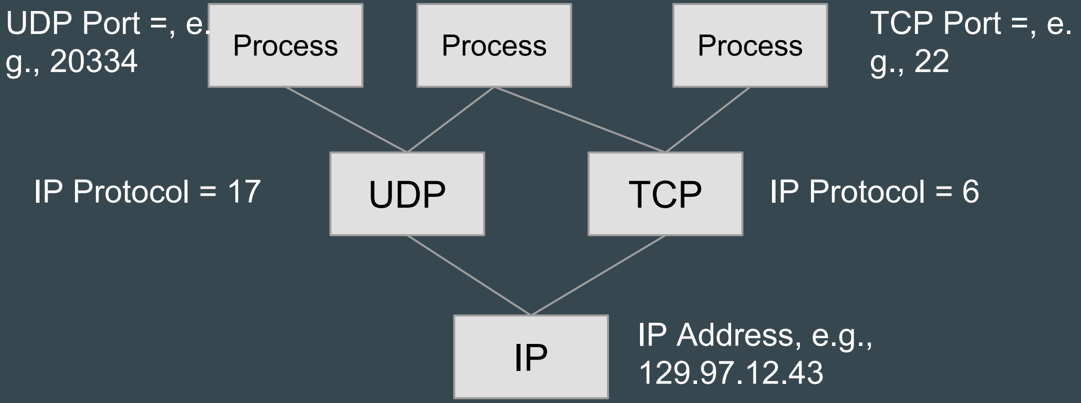
Focus on UDP & TCP over IP

- UDP & TCP are transport-layer protocols
- Over IP, which is a network-layer protocol



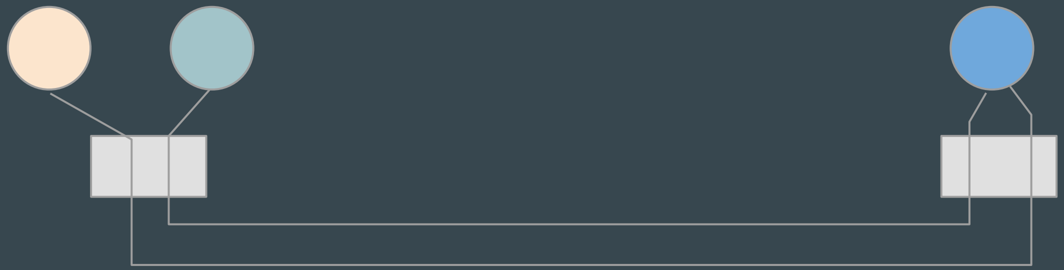
- UDP & TCP multiplexed over IP.
- Multiple processes multiplexed over each of UDP & TCP.

Multiplexing



The “5-tuple”

- On an (the) Internet, a connection or association is identified uniquely by the 5-tuple:
    - ⟨source-ip-address, source-port, destination-ip-address, destination-port, protocol⟩
    - E.g., ⟨129.197.2.13, 20334, 216.58.199.14, 80, 6⟩
    - E.g., ⟨129.197.2.13, 50000, 216.58.199.14, 80, 6⟩



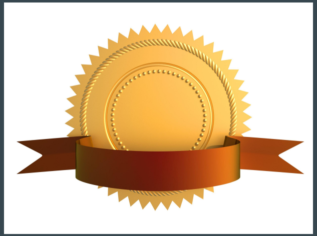
The software view

- OS/library for UDP/TCP
- Applications can be written on top of UDP/TCP
- Client (initiator) - server (responder) paradigm.

The socket API

- POSIX standard API for UDP/TCP applications
- (Does not necessarily mean that it's a great API.)
- Essential calls:

  - *socket()*
  - *listen()*
  - *accept()*
  - *send()*
  - *sendto()*
  - *bind()*
  - *connect()*
  - *receive()*
  - *sendto()*
  - *recvfrom()*



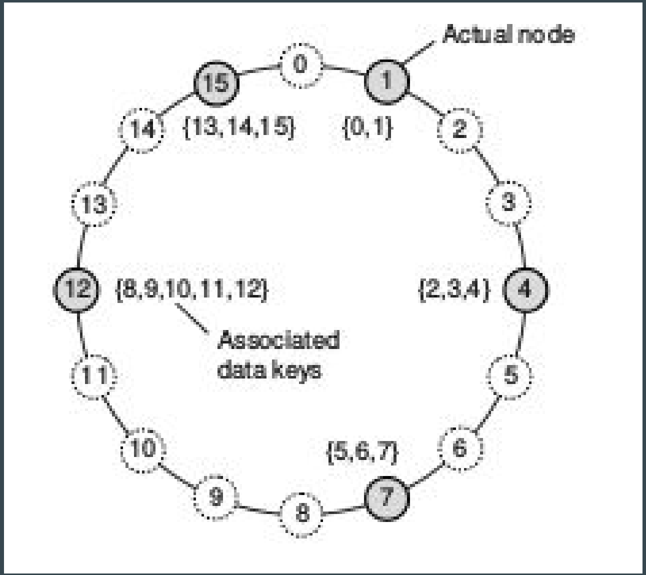
UDP, TCP Applications →

Overlay Networking - p2p over TCP,UDP/IP

- Want a p2p network that maintains content.
- How to distribute content amongst peers?
- A client may contact some peer and want to lookup content.

The Chord DHT

- *m*-bit *key* unique for every peer and piece of content.
- Define: *succ(k)* for any key *k* is peer that exists with smallest *id* ≥ *k*.
- Content with key *k* hosted by *succ(k)*.

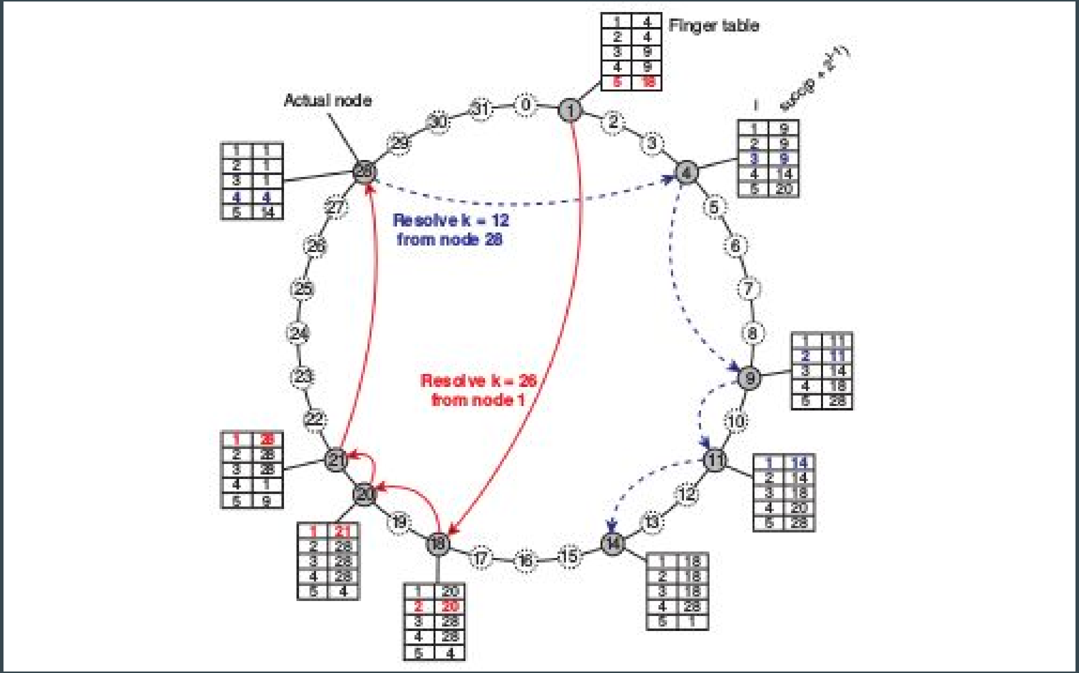

Actual node
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
{13,14,15}
{0,1}
{2,3,4}
{8,9,10,11,12}
Associated data keys
{5,6,7}

lookup(k)

- We may want to *lookup(k)* at any peer.
- Query is routed to *succ(k)*. How?

The Chord Approach

- Maintain a “finger table” (routing table) at Peer *p*, $FT_p[\,]$.
- *m* entries
- $FT_p[i] = succ(p + 2^{i-1})$, for all $i \in [1,m]$


Finger table
i
succ(p + 2^(i-1))
Actual node
Resolve k = 12
from node 28
Resolve k = 26
from node 1

Artwork credit

- … (to be completed)