582615 - Overlay and P2P Networks, Spring 2015

Exercise 1 answers

Assignment 1

Layers
Application
Routing / Topology
TCP / UDP
IPv4 / IPv6
(Ethernet etc.)

- There are more than one view for what constitutes an overlay, typically at least these:
    - Application / Service layer: Protocols or messages of any software that uses the overlay.
    - Routing / Topology / Security layer: Logical topology of the overlay, routing and forwarding messages among nodes, possibly joining and parting the overlay network. Possibly encryption. Depending on the overlay may provide services such as NAT traversal.
        - Depends on overlay.
        - In the exercise session presentation, these were called “Essential services”.
    - Network: The underlay, underlying network, this usually is the TCP/IP stack,
- Overlays by definition assume the presence of an underlay, typically the TCP/IP stack. In relation to TCP/IP, the overlay stack usually plays the role of an application level protocol. From the point of view of the overlay stack, the TCP/IP stack usually acts as the network layer.
- Overlays often do provide similar services as the underlay; the difference is context.
    - Routing and forwarding: e.g. path (ordered list of nodes) through the Tor network vs. the same path through the internet. One hop on the logical Tor path involves many hops through the physical internet path. On both layers each hop requires a routing decision relevant to that layer, and forwarding of the data to the next step.

Assignment 2

- Almost any kind of system can use an overlay, actual benefit depends on which application and which overlay.
- Overlays provice an easy way to extend functionality of network or application, and as a reusable component, can simplify application development.
- Overlays can help overcome limitations of underlay network, (e.g. bring support for mobility).
- Often helps with scaling a service when resources need to be added or removed.

Trade-offs

- Plus
    - Simplify application design, features don't need to be built into the applications
    - Possibly better interoperability
- Minus
    - Added components increase system's overall complexity; more places for bugs.
    - More layers, more communication, more netwrok and computation overhead.
    - The logical topology may not be optimal for the physical topology.
    - Overlays give away some control (as does anything that does things / makes decisions for you).

Changing the internet

- Replacing the established protocols is difficult.
    - Perspective: IPv6 (RFC 2460) was introduced in 1998. Today globally less than 20% of ASes (Autonomous System) announce an IPv6 prefix. (RIPE NCC).
- Changes that are good for one purpose may be bad for other purposes.
- A lot of stuff in core networks is done in hardware, changes would require replacements.
    - May not be justified from the operators' point of view.

Assignment 3

- Main difference is that in P2P, the Peers (“clients”) take over some or all responsibilities of the server. Instead of a centralized server, or in addition to one, there is a network of peers, with the computation, memory and networking resources that collectively are likely to exceed those of a centralized server.
- Strengths
    - Server load is distributed to the edges; less need for expensive hardware.
    - Bandwidth comes from the edges, less need for fast connections for servers.
    - Single point of failure (server) can be removed. May be able to tolerate server crashes, blocked connections to server etc.
- Weaknesses
    - Some operators try to block P2P systems, technically or through user contracts (esp. in mobile broadband).
    - NAT devices cause problems.
    - Privacy: Peers, i.e. users have direct connections with each other; users may learn about each other things that traditionally only the server would. http://www.cs.bham.ac.uk/~tpc/Papers/P2PSecComm2012. pdf
- NAT (Network Address Translation)
    - Hosts behind a NAT device usually share a public IP address.
    - Incoming connections fail if ports are not forwarded
        - Solutions: UPNP, PCP, hole punching, relaying, manually forwarding ports to selected host.
    - Protocols with public IP (plain or hashed) as identifier; identifiers collide if many hosts behind same NAT device use the protocol.
        - Solutions: Different choice of identifier.
    - Hosts behind a NAT device may not know their public IP -> private IPs in protocol messages.
        - Solutions: STUN, ICE,… or designing the protocol so that there are no IP addresses in messages.

Assignment 4

- Rarest first algorithm:
    - Used most of the time during download.
    - Maintain a list of rarest pieces, always request them first to keep them from disappearing from the network.
- The policies
    - Random first policy
        - Need something to upload before download can really start, used for bootstrapping the download.
        - Used for the first four pieces.
        - Ignores rarest first algorithm.
    - Strict priority policy (all blocks of same piece first)
        - When a block of a piece is downloaded, request the rest of the pieces with highest priority
        - Used at all times during the download.
    - End game mode
        - Used after all pieces are requested.
        - Flood request for pieces to all who have it. Cancel requests as pieces arrive.
        - Used to finish the download quickly.
- Fairness of original Choke algorithm
    - Fairness is based on the assumption that all nodes contribute. This turns out to not always be true.
    - Downloaders: Fast peers get priority, for it was assumed that fast peers would also disseminate the content faster. A leech with a fast connection, regardless of whether they help disseminating the content or not, would get priority over slower leeches. Presence of fast, but non-contributing leeches would degrade performance for slow leeches.
    - Uploaders: The non-contributing leeches would not help with disseminating the content; the seed ends up having to upload all the pieces again to other peers by itself; It takes longer to disseminate the content.
    - Overall: The overall throughput of the network would suffer if there are freeriders with fast connections.
- Seed State Modification
    - Give all downloaders the same constant duration of service, regardless of speed.
    - Tolerates freeloaders, but doesn’t let them dominate. Overall more fair.

Turbo challenge

Advantage

When using a DHT, the responsibilities of the tracker get distributed among the nodes. This comes with at least the following advantages:

- Fault tolerance: The single point of failure (the tracker) can be removed.
- Scalability: The demand for tracker is distributed.
- Availability: One can still access the torrent if traffic to the tracker is blocked.

Peer discovery

The DHT based peer discovery is a separate process from downloading torrents over the BitTorrent protocol. The DHT carries peer information for multiple different torrents at the same time; the torrents' *info_hash*es being keys and peer connection information being the values.

When joining the DHT, a node chooses a random globally unique identifier (*Node ID*) from the same namespace as the *info_hash*es used to identify torrents. It also bootstraps itself with a short list of connection information for other nodes participating in the DHT. The bootstrapping can be done through metainfo in the torrent file if included, through a central bootstrapping server or other implementation specific mechanism. After bootstrapping each node carries a routing table with *Node ID*'s mapping to node IPs and ports.

The *Node ID*s and *info_hash*es are all from the same namespace, this makes it possible to use a XOR metric for their distance, which is then used to decide which nodes should carry information about peers for which torrents. When a has joined the DHT and wants to find peers for its torrent download, it takes the *exclusive or* (xor) operation of the torrent's *info_hash* and its peer nodes *Node ID*'s, then sends the request to the ones it found to be closest to the *info_hash* (lowest value output of the xor operation).

The node then responds with a list of peers if it knows any, or performs a similar comparison and responds with a list of DHT nodes that are closer to the *info_hash*. The routing table should contain more addresses that are close to the node's own address for it to have a better chance of providing others with relevant node contact information.

If the requesting node receives a list of BitTorrent peers, it tries to peer with them. If the response was a list of DHT nodes, it will try asking them for peers. This sort of lookup is *iterative*, in contrast to *recursive* where the responding node would forward the request to the closer nodes.

Once the search for peers is finished, the node inserts itself to the peer list of the torrent so that others requesting for peers to that torrent later would find it. To authorize adding to the list, the node must present a token that it has received from the node responsible for the list earlier.

As to which pieces a node has; this information is exchanged during the handshake while peering. It can also be announced to the node closest by XOR-metric to the torrent in question, but this is implementation specific.