582615 - Overlay and P2P Networks, Spring 2015

Exercise 2 answers

Considering the following points in the answer granted points in the assignment.

Assignment 1 - Freenet

How does freenet fetch files?

Resources on Freenet are identified by Globally Unique Identifiers (GUID) which can be either Signed-Subspace Keys (SSK) for identifying pointers to content, or Content-Hash Keys (CHK) for identifying content itself. Every node holds lists of GUIDs as part of their routing table identifying which neighbours are likely to know about which resources.

Upon receiving a request for some CHK, the node will first check its own data store. If it did not hold the data, the node will forward the request to the node that has the closest key to the one requested. That node will then follow the same procedure. This is essentially a depth-first search starting from self.

If a node that gets the request is already in the search chain, it will bounce the request back to the node that it got it from. When a forwarded request bounces back, the node will try forwarding it to the node holding the second closest key and so on until the forwarding table is exhausted. Exhausting the forwarding table also causes the request to be bounced back to the previous node.

Once the node that does hold the data is found, it will return the file back the same route that the request traveled, marking itself as the holder of the data. The marking is to help the nodes along the path update their routing tables. The nodes along the path must forward the file back upstream, but they may additionally cache the file and/or change the reply message marking themselves as the file holder. The additional tasks are to protect the identity of the real data holder.

Opennet vs Darknet

Opennet is the ‘regular’ Freenet, where a client starts itself with the help of a list of known hosts called seed nodes. Connections are formed to unknown peers found on public directory listings. Opennet is the default way to start using Freenet.

In darknet, the client peers only with nodes whose details the user has inserted manually into the system. It is expected that the user personally knows and trusts the other users with whom they choose to pair.

The difference is the way in which closest neighbours are selected. The content that users receive is the same in both modes. Darknet is considered more secure, because it surrounds the user with only trusted peers, and the protection of anonymity in Freenet is better the further you are from the host.

Assignment 2 - Network modelling

Power law

- Relationship between two variables where the other varies as a power of the other.
    - e.g. When looking at nodes and their neighbours, there are a few nodes with many neighbours and a lot of nodes with very few neighbours, the degree distribution may follow a power law.
- Also ok to say this through reference to log-log scale.

Small-world

- A small-world network is a network where the shortest path between any two nodes tends to be short.
- There are cliques; some nodes have more neighbors than the others.
    - Clique ~ a hub + its neighbors; can be other hubs.
- Number of steps shows roughly logarithmic growth to the number of nodes.

Scale-free

- In a scale-free network the degree distribution (~number of neighbors the nodes have) of the network follows a power law.
    - The more there are nodes, the more there are nodes with many neighbours.

Common/difference

- There is a power law.
- In both networks, average shortest lengths between nodes is short.
- Focus on the definition is different
    - In small-world, the focus is on path network diameter / path lengths.
    - In scale-free, the focus is on degree / numbers of neighbours.
- Scale-free is sort of an extreme case of a small-world network.
- Not all small world networks are scale free.

Resiliency against random node failures

- If the degree distribution follows a power law, there are a lot of nodes with a few connections, and a few nodes with a lot of connections. If we choose a node at random, we are likely to select a node with only a few connections. likely leaving most of the network unaffected.

Assignment 3 - Consistent hashing

How does consistent hashing work?

- The hash table indexing is modeled as a ring
    - Fixed address space [0,1[
- The ring is partitioned into buckets.
- Positions for buckets are chosen randomly (by a balanced random function) and the buckets are assigned to nodes.
- Positions for objects are chosen by a balanced hashing function.
- The key idea is to distribute the objects to buckets evenly.
    - evenly != strictly equally.

Joining and parting

- When a node joins, it is given one or more buckets at random points on the ring.
    - Whether it chooses the positions itself or they are elected is an implementation detail.
- It receives keys from the clockwise next buckets for its buckets.
- When a node parts, its buckets become part of the clockwise next buckets.
- Since the address space is fixed, and keys always hash to the same address, only the parts where the buckets or nodes change need to transfer objects.

Hash function balance

- A well balanced hash function distributes the buckets and keys evenly among the DHT.
- A poorly balanced hash function will burden some buckets significantly more than others, causing most of the work to be done by a small set of nodes.
- An unbalanced hash table is not as efficient as a well balanced one.

Replication

There are a couple of strategies:

- Place replicas in the N buckets following the key's own bucket.
    - If the key's owner parts, a replica is already with the node gaining the responsibility.
- Add a known suffix to the key of replicas
    - insert(key, value)
    - insert(key+1, value)
    - insert(key+2, value)
- Make more than one node be responsible for the same bucket, each replicating it.

note

- Node = a computer participating in the DHT
- Bucket = a partition of the object space (a section on the ring)
- A node can carry one or more buckets, depends on configuration.

Assignment 4 - Bloom filter

Difference to hash table

- Use
    - Regular hash tables are used to map a key to a value.
    - Bloom filters are used to determine whether a key might be found in a set, not to store the values.
    - Bloom filters can be used to determine, in constant time, whether it's worth looking for a key from a larger data structure.
- Operation
    - (Regular hash table examples simplified, implementations need strategies for special cases)
    - Initial
        - Hash table: Empty 2-dimensional array.
        - Bloom: Bit array full of 0's.
    - Insertion:
        - Hash table: Map the key to exactly one index, store value there.
        - Bloom: Map the key to a set of indexes, store the bit 1 in each.
    - Lookup:
        - Hash table: Map the key to exactly one index, return value if found.
        - Bloom: Map the key to a set of indexes, return true if they're all 1.
    - Deletion:
        - Hash table: Map the key to exactly one index, clear the value.
        - Bloom: Not possible. Breaks lookups for keys where at least one function collides.

Results

- Bloom filters are a lot smaller than the real data structures, distributing metainformation through Bloom filters is cheap.
- The nature of the false results is known: Bloom filters may return false positives, but never false negatives.
    - They can be used for quickly determining whether the key might be found, or can definitely not be found in a set.
    - Quickly find out if it's worth even trying to query the data set itself.

Insertions and lookups

- Bloom filters consist of a bit array and a set of hash functions.
- Each key is given to a set of hash functions, that each map the key to an index in the array.
- Insertions are done by setting the bit at all mapped indexes to 1
- Lookups are done by checking whether all the mapped indexes are set to 1

Removal of elements

- The standard Bloom filter does not support removal of elements; the whole filter would have to be reconstructed without the element to be removed. Setting bits to 0 in the key's positions would introduce false negatives if there has ever been a hash collision during insertion.

Turbo challenge

Anonymity

- Encrypting the messages before they leave the computer, an eavesdropper could not access the content nor the target of the messages
- Traversing the encrypted message along a path of Tor nodes hides the original source of the message from the target.
- Onion routing hides the original source from the Tor nodes, and target from all but the exit node.

Onion routing

- The initiator node chooses a random set of nodes it knows about and order for them; this order is the path and the last node is the exit node.
- It then uses public key cryptography to negotiate a session key with the first node (entry node).
- Second it asks the entry node to negotiate a session key with the second node.
- This repeats all the way up to the last node (exit node).
- It then encrypts the message in multiple layers, starting from the key of the exit node, ending with the key of the entry node.
- When the first node gets the message, it decrypts the first layer of the and forwards the message to the next node; this repeats until the exit node.
- The exit node decrypts the last layer of encryption and forwards the plain text message to the internet.
- When a response comes back, the exit node encrypts it with the session key, and the response travels back the same path, gaining a new layer of encryption on each step.
- When it arrives at the initator, it decrypts all the layers with the session keys and reads the response.

Hidden services

- Reside within the Tor network, don't go through an exit node.
- Addressed by their onion address.
- Side-effect of Onion routing: end-to-end encryption

Anonymity = security?

- Tor does not encrypt the communication between the exit node and the target.
    - Sending identifiable information (e.g. usernames / passwords) without securing it can compromise anonymity.
- An application that does things both inside and outside the Tor network can be revealed if actions inside the Tor network trigger actions outside.
- Someone in control of an Autonomous System can apply statistical methods if both ends of the Tor path are within that AS.
- Improperly configured hidden services (especially ones available on public www as well) can compromise anonymity.