Distributed Backup Service: Enhancements

Chunk backup subprotocol

The project specification proposes the following actions to be taken when a PUTCHUNK message is received:

1. Store the chunk
2. Wait for a random interval uniformly distributed between 0 and 400 ms
3. Send STORED message

Our enhancement does not require any additional special messages, and therefore is inter operable with any other implementations of the protocol:

```
wait for a random interval uniformly distributed between 0 and 400 ms;

meanwhile, save the number of received STORED messages related to the chunk being backed up

if (that number < desired replication degree) {
  store the chunk;
  send STORED message;
}
```

This enhancement:

- Ensures the desired replication degree almost every time
- Does not deplete the backup space rapidly

Chunk restore protocol

To do.

File deletion subprotocol

To do.

Space reclaiming subprotocol

The project specification proposes the following actions to be taken when a REMOVED message is received:

1. Update local count of peers backing up the chunk (chunk mirrors)
2. If that count drops below the desired replication degree of that chunk
    - Wait for a random interval uniformly distributed between 0 and 400 ms
    - If a PUTCHUNK message has NOT been received meanwhile
        - Start chunk backup subprotocol

This implementation has the following flaw:

If a peer fails during the chunk backup subprotocol step, the replication degree of the file chunk may be lower than desired.

Our enhancement does not require any additional special messages, and therefore is inter operable with any other implementations of the protocol:

When a peer receives a REMOVED message:

```
if (peer is backing up a copy of that chunk) {
  update available mirrors of chunk;

    if (new replication degree < desired) {
        wait for a random interval uniformly distributed between 0 and 400 ms;

        meanwhile, save number of received PUTCHUNK messages;

        if (no PUTCHUNK message was registered meanwhile)
            start backup sub-protocol of that chunk;
}
```

Meanwhile, the peer who sent the **REMOVED** message does the following:

```
listen PUTCHUNKs for 500ms;

if (no PUTCHUNK was listened) {
    ask for the chunk to be deleted

    listen for CHUNKs for 500ms;

    if (no CHUNK is listened)
        none of the peers has the chunk.
}

if (at least one PUTCHUNK was listened **OR** none of the peers has the chunk) {
    repeat 5 times, double waiting time every retry {
        wait for 500ms;

        meanwhile, listen STORED messages;

        if (STORED message received)
            stop repeating this block;
    }
}

finally, delete stored chunk;
```