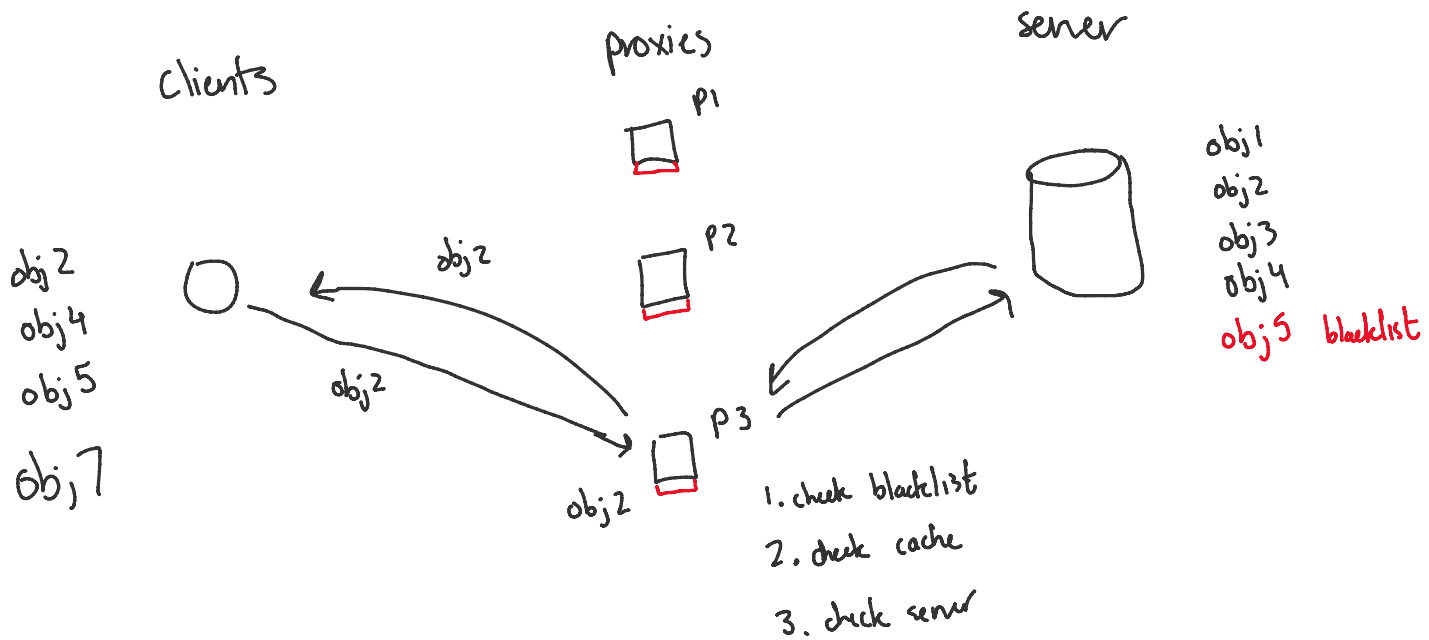
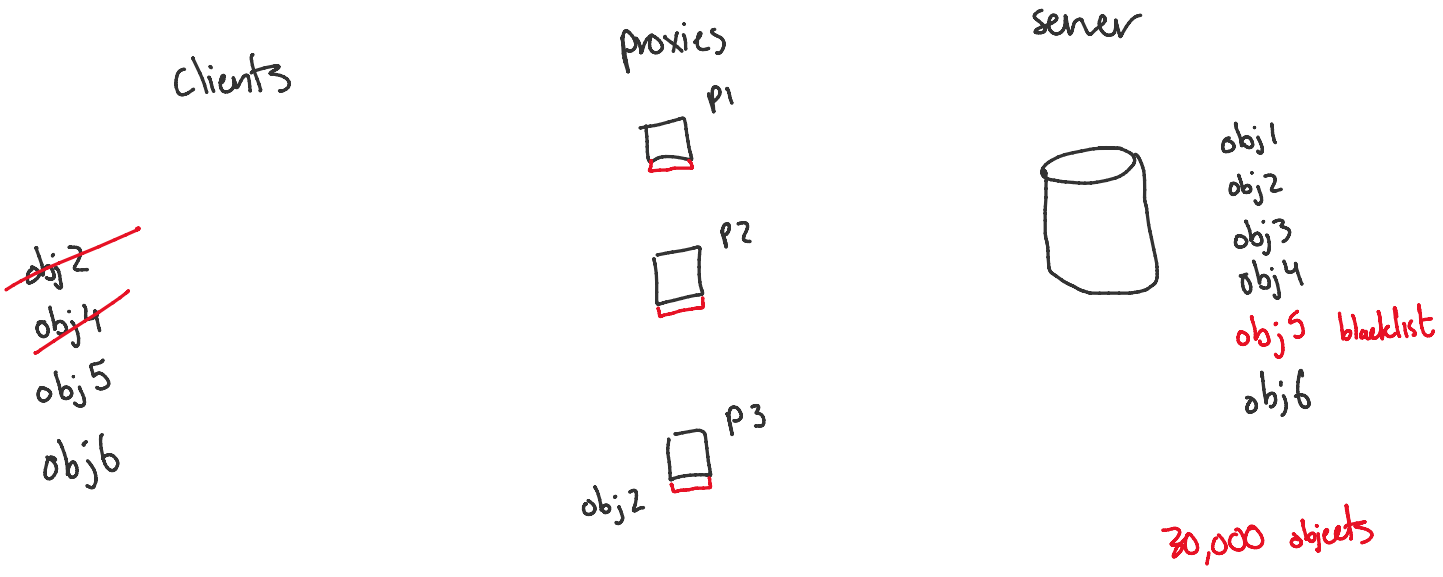
Final Project Design 2

Wednesday, November 18, 2020 7:03 PM





HRW

murmur hash

n elements

Bloom Filter

- probabilistic data structure
- confirms existence
- there are false positives
- no false negatives

obj1 obj2 obj3

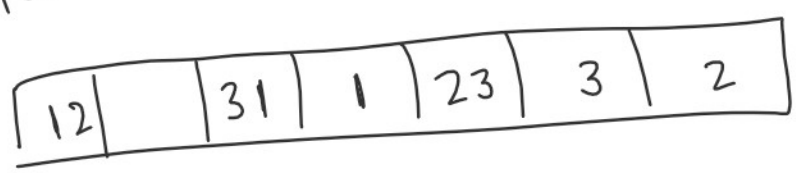
k hashes

h1(x)
h2(x)
h3(x)

obj4

8 bits

1 bit 1 bit 1 bit

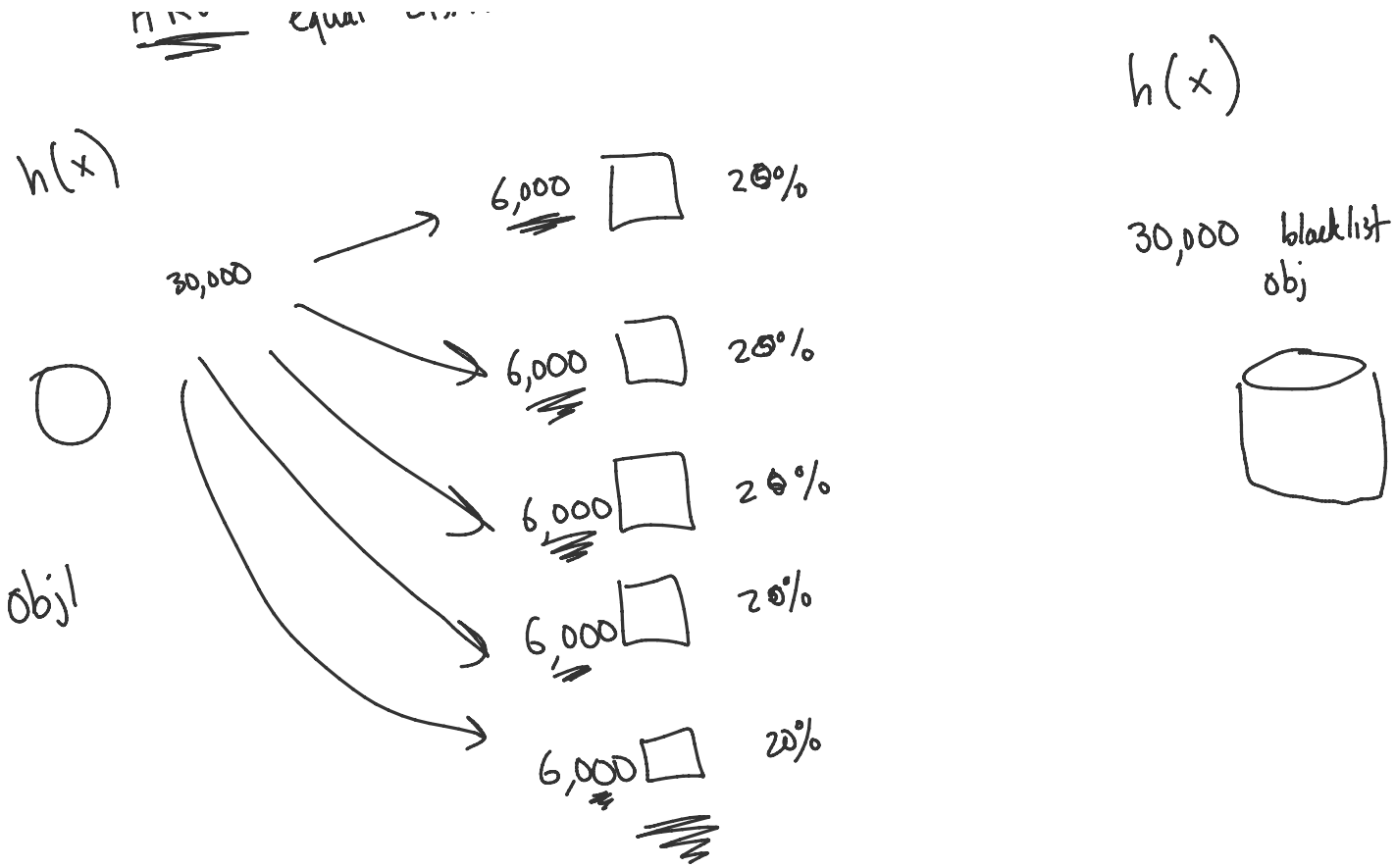


- for blacklist elements
- at most 30,000 elements : n
- 5 hashes : k
- at most 1% hit rate : [PR]

Size of m: 295,000 bits → all proxies

HRW equal distribution

h(x)



60,000 bits, or 7500 bytes per proxy

- TLS
- Packet design
- HRW
- Bloom Filter

- Servers
- Clients
- Proxies

- Mummur

The proxy application should be executed as follows:

```
your_application_name -port portnumber -servername:serverportnumber
```

The server application should be executed as follows:

```
your_application_name -port portnumber
```

↳ how to incorporate names

class:

application => used by client only

↳ enforce TLS

↳ enforce certification

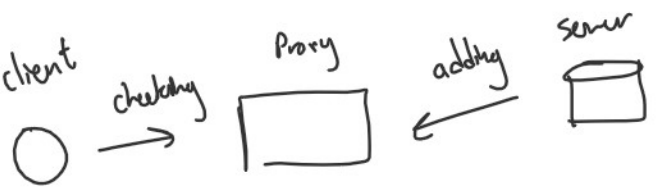
↳ enforce
↳ enforce certification
↳ allow user to add input file
↳ small delay between
↳ contacting proxies
- HRW

hard code proxy names

HRW → inputs: obj-name

output: best proxy-name

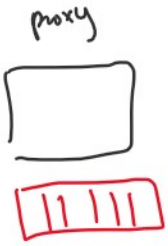
Bloom Filter → class



obj1
obj2
obj3
obj4
obj5

1)

blacklist → pr1, pr2



6,000 obj

bloom filter initialize (server):

inputs: char** blacklist, int size, self
(null terminated) (blacklist)

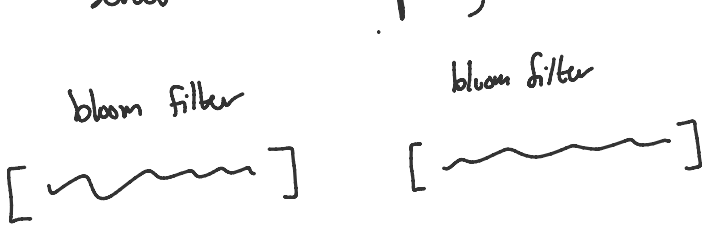
1) dest fill with zeroes
2) add all objects to dest with hashes

outputs: none

bloom filter accept (proxy):

inputs: char* recvBuffer, self
- copy over buffer

outputs: None



bloom filter check (proxy):

input: char * obj_name

output: return 1 if on blacklist

Output: return 1 if on blacklist
return 0 if not

1) HRW 11/23
2) Bloom Filter 11/23

1) TLS / Packet Design 11/23
2) Basic server/client architecture: focus on sys args 11/23