Freenet: Clarke, Miller, Hong, Sandberg, Wiley [Clarke00a]

CSci551: Computer Networks
SP2006 Thursday Section
John Heidemann

Peer-to-peer Systems Intro

- why p2p?
  - many reasons
  - Freenet: anonymity
  - Napster: how to easily find content
  - Bittorrent: exploiting parallelism in download process
  - Chord: binary search around ring
- things to look for in p2p systems
  - search (both *name to key* and *key to location*), update, redundancy/fault-tolerance

Key ideas

- distributed storage system
- anonymity
  - search/routing indirectly through other peers
  - data can be stored anywhere
    - difficult to censor or stop data in the net
    - no records of who posted data: difficult to find poster
  - keep data encrypted
    - you can't tell what data you have
- routing via "node chains"
  - and optimizing routing via "hill climbing" approach
- replicate files on retrieval (to improve fault tolerance and performance)

Preliminaries: the Politics

- Freenet (more than other protocols) has an explicit political goal
  - distribute data (any data)
  - anonymously
- builds on prior work in anonymous e-mail

Freenet Components: GUIDs

- GUIDs
  - » globally unique identifiers: SHA-1 hashes of something, used to locate the key
  - goal: have short identifier to stand in for filename or keyword or contents

Freenet Components: public key pairs

- files have a *contents*, and a *public-key pair*, and one or more *description strings*
  - content is Declaration of Independence, key xxx, string: politics/doi
    - store the SSK under h(politics/doi), that then points to h(contents)
    - store the CHK under h(contents)
- why?
  - "allow users to create their own space"
  - need some mechanisms to allow someone to make changes to their data
    - yet don't want to know who they are
    - authenticate to some anonymous user X, not a name like "John Smith"
    - compare to other systems which add a PKI: public key infrastructure that maps public keys through a chain of trust back to known third party

Freenet Components: CHK

- Content Hash Keys (CHK): like file system file contents
  - CHK=sha1(file-data)
  - given a CHK, we can search for the file data
  - file data is signed with the private half of the public key
    - so anyone with the public have can check it
    - but only someone with the private half can re-sign it
- why?
  - can identify different files, uniquely
  - need short identifier to use in search for file contents

Freenet Components: SSK

- Signed Subspace Keys (SSK): like file system directory entry
  - SSK=sha1(description-string); also includes sha1(sha1(public-key),sha1(description-string))
  - to find a file, must know the public-key and the description string
- why?
  - tagging the file contents with some name (the description string)
  - useful for finding data by the description string
    - given description string, easy to compute the ssk

Basic Idea: Finding Data

- routing (not a standard term for this)
  - generate a key (SSK) from the filename
    - it's just a hash, a "random" 160-bit number
    - gives the CHK
  - find data by looking up CHK in network
- search/discovery
  - finding the filename in the first place
  - out-of-band, or maybe do automatic indexing, or sharing names public

Basic Idea: Routing

- throw data into a mesh of nodes
- each node has a routing table listing which neighbors have which keys
- route queries towards keys
- encourage locality in where the keys are stored; how:
  - replicate data as it is returned to the user
- performance
  - worst case: O(n) where n is the number of nodes
  - average case: don't have strong answer, but their simulations say with 10k nodes, only 8 hop search most of the time
  - locality: hope is that hash values promote locatility around similar values

Basic Idea: Anonymity

- when propagating requests, add randomness to obscure sender/receiver
  - examples: each node pretends to be the original requests, nodes can tweak TTLs
  - where have we seen this before? xxx
- data is encrypted and stored by key, so node owner doesn't know contents
- updates are hard
  - use public-key encryption to allow only owner to update

Naming

- first, strings that map to hashses
  - text/philosophy/sun-tsu/art-of-war => sha-160 hash 0x12838482
- but how do we know *which* strings?
  - could be Prose:Philosophy:Chinese:Sun Tsu:Art of War => 0x8348234f
- and since the SSK includes the public-key, where do you get that?!?
- and how do we browse?
  - what other Philosophy texts?
  - what's the equivalent of Google or Yahoo?
- they don't have any real answer

Updating

- how do we update data in place?
  - can't just replace data, because that allows denial of service
  - yet need to update data in place (ex. to maintain directories of keys)
- single user
  - can use public key cryptography and indirection
  - encryption/security details: see CSci555

Compare to Other Peer Systems

- Napster had a central database, it's distributed
  - Kazaa and Morpheous too (right?)
- Gnutella?
- others have better
  - search
  - user interface?

Does it work?

- Not clear if Freenet scales…
  - with sparse keyspace, how much flooding?
- Vulnerable to DoS attacks…
  - record companies putting songs with 15s of music and then a raspberry
  - *no real way to stop this; why?*
    - xxx
- Not clear that search is sufficient..
- But very interesting design point

Comparing to Other p2p Systems: FreeNet

- search:
  - finding a name: no real directory system
  - finding a key: hill climing algorithm
- update:
  - insert: just like search
  - update in place: uses public key stuff
- redundancy: many copies of file X
- other features: anonymity

Other questions/observations?

- hash complexity?
  - must make one pass over data
- can you turn off anonymity/encryption?
  - no

Example File

- given name: class15.ppt
- map to SSK: sha1("class15.ppt") = 0x234
- locate SSK: find 0x234
  - does the multi-step search
  - replicates the contents on the way back
  - returns SSK contents:
    - publickey of poster X1
    - CHK = h(contents of class15.ppt) = 0x667
- locate CHK: find 0x667
- get data: $\{\text{contents of class15.ppt}\}_{X1}^{-1}$
- then use public key X1 to decrypt data