



Fall 2009

Dr. Kevin Hamlen


CS 4485: CS PROJECT



Stage 3:
Due NEXT WEEK



New Features

- Keywords
 - Associate
 - Unassociate
 - Lookup
 - Leave
 - graceful exits
 - non-graceful exit recovery (forwarding failures)
- 



STAGE 4: The Finished Product!

Stage 4 Requirements

- Make it production-level
 - clean, user-friendly GUI
 - easy to configure for a typical machine
 - Ex: Make the port number user-configurable so that users behind NAT's can use it.
 - no crashes!
 - helpful error messages when something goes wrong
- User manual
 - readable by non-computer scientists
 - explains all GUI functions and what they do
 - make it online accessible (e.g., webpage or Windows help file)
- Technical manual
 - readable by programmers or other computer experts
 - explains how to configure & run the test suite, and what it does
 - documents the programmer API used by the test suite / GUI
 - can exist in any form (paper, Word, webpage, etc.)

Stage 4 Requirements (cont)

- Implement at least ONE of the following:
 - **Option 1:** k-replication (standard vs. consecutive)
 - **Option 2:** message-dropping defense (probabilistic forwarding vs. multiple overlays)
 - **Option 3:** voting + boolean expression search
 - **Option 4:** get your product to interoperate with another team's product written in a DIFFERENT language

Stage 4 Submission

- Stage 4 Demo
 - My office (ECSS 3.704) on Friday, Dec 11th
 - No documentation required for demo
- Final Submission
 - Due Tuesday, Dec 15th by 1:00pm (note changed date!)
 - Submit as TWO CD's/DVD's containing:
 - source code (including project files, etc.)
 - documentation (user manual & technical manual)
 - binaries (if non-Unix)
 - installation instructions & manifest (README.txt)
 - Any paper material (tech manual, final report, last will and testament, etc.)
 - Be sure to submit TWO copies of everything (one for me, one for the TA)



Extra Stage 4 features
(i.e., the fun part)

Option 1: k-replication

- Instead of having just one key-holder and one keyword-holder for each file/keyword, have k of them.
- Modify lookup, publish, and retract procedures to contact all k
- Use majority decision in lookups
 - Example: if 4/5 say a keyword is not associated with a given file, but the 5th does, the “nots” win.
- Implement BOTH of the following separately (make it configurable):
 - k replicas are chosen via family of k hash functions
 - k replicas occupy consecutive ring positions

k-replication (cont)

- Standard k-replication
 - $H_i(\text{input}) = H(\text{input}:i)$ (for each i in $[1..k]$)
 - k is a fixed compile-time parameter
 - possible that multiple i 's hash to the same peer when the network is small, but very unlikely when the network is large
- Consecutive k-replication
 - all peers whose id-ranges contain an id "between" $H(\text{file_id})$ and $(H(\text{file_id}) + R) \bmod 2^{32}$ will be key-holders
 - R is a fixed compile-time parameter (less than 2^{32})
 - New forwarding operation: Deliver this message to all peers whose id-ranges contain a number "between" X and Y
- Report comparison
 - compare average number of messages / message-hops for different k and different R
 - provide graphs comparing the two approaches

Option 2: Message-dropping

- Probabilistic Routing
 - Forward message to closest finger table peer with probability p , to next-closest with probability $(1-p)$
- Multiple Overlays
 - Each peer gets k identifiers instead of just one
 - Each peer has k finger tables, one for each ring
 - Each peer has k key-databases, one for each ring
 - Publishes and associations use ALL k rings
 - Lookups may use ANY of the k rings
- Compare the approaches
 - Have a percentage x of peers drop all messages
 - Run the test suite
 - On average, how many messages get delivered?
 - What's the best value for p when using probabilistic routing?
 - What value of k is required to match the success rate for the best p ?

Option 3: Voting+BoolSearch

- Allow clients to vote on keyword associations
 - keyword-holder tracks two counters x and y for each keyword-file pair in its keyword database
 - each associate increments x and y
 - each unassociate increments only y
 - may unassociate without associating to cast a negative vote
 - association favorability rating equals x/y
- Prevent (some) multi-voting
 - Policy: at most one vote per IP per keyword per file
 - Disallows some legal voting (many users from one IP)
 - Vulnerable to Sybil attacks, IP spoofing, etc.
 - Could fix this by having a more robust authentication system (e.g., digital id's), but that's beyond our scope

Option 3 (cont): Preventing Multi-voting

- Need to remember a list of IP's of voters for each file-keyword pair
 - Could require keyword-holders to track it, but that might be prohibitively expensive for some peers
 - Instead, the current vote for/against keyword w for file with hash f from IP i will be remembered by the peer whose id is closest to $H(w:f:i)$ rounded down.
- Voting protocol
 - $\text{voter} \rightarrow \text{id}_{H(w:f)}$: (Un)associate (w,f)
 - $\text{id}_{H(w:f)} \rightarrow \text{id}_{H(w:f:i)}$: IP_{voter} votes for/against (w,f) .
 - $\text{id}_{H(w:f:i)} \rightarrow \text{id}_{H(w:f)}$: vote change: (yes/no/none) \rightarrow (yes/no/none)
 - $\text{id}_{H(w:f)}$: increase or decrease x , possibly increase y

Option 3 (cont)

- Search expression can ask for matches whose voter tallies exceed some threshold
 - Example: "reggae:70" searches for songs such that at least 70% of voters think it's reggae
 - Default is ":50"
- Add boolean search capabilities:
 - (reggae:70 OR blues) AND NOT rap:20
 - invent your own syntax if desired, but support conjunction, disjunction, negation, and grouping (parentheses)

Option 4: Interoperation

- Find another team whose implementation is in a DIFFERENT language than yours
- Collaborate to get your two clients to communicate on the same network
 - all features should be supported
- Do NOT merge the implementations! Clients should remain in their original languages, with their original GUI's, etc.