Gnutella (cont)

Query
  ファイルの検索要求 (search for files)

  flooding による転送
  (routed by flooding)

  payload: >2 byte
    Minimum Speed (2)
    Search criteria (>0)

  Minimum Speed (KB/sec)
    ファイル要求側が要求する回線速度
    (the line speed required by the requesting servent)

  Search criteria
    null ('\0') terminated search string
    検索文字列の意味は実装に任されている
    (the semantics of the file name matching is left to the implementation)
      部分文字列マッチ (substring matching)
      正規表現 (regular expression matching)

QueryHit
  Query に対する応答 (reply for a Query)
    descriptor header 内の descriptor id field は
      Query の descriptor id field に一致させる
      (descriptor id of QueryHit == that of Query)
    Query の来た経路を逆にたどって転送
    (routed along the route of Query reversely)

  payload >27 byte
    Number of Hits (1)
    Port (2)
    IP Address (4)
    Speed (4)
    Result set (>0)
    Servent Identifier (16)

  Number of Hits
    検索結果のヒット件数

  Port, IP Address
    servent のデータ転送接続受け入れアドレス
    (used for file transfer connection)

  Speed (KB/sec)
    servent の回線速度 (line speed offered by the servent)

  Servent Identifier
    servent を識別
      いつも同じ値を使う (always same value for a servent)
    Push descriptor で使用(後述 explained later)

  Result (>8)
    File Index (4)
    File Size (4)
    File Name (>0)

  File Index
    ファイル提供 servent 側でのファイルの番号
    (index in the servent offering the files)
  File Size (bytes)
  File Name
    double-nul (i.e. 0x0000) terminated name of the file

Push

```
  後述 (explained later)
-------------------------------------------------------------
File Download
  QueryHit に含まれるファイルを入手したい
  (transfer a file contained in the QueryHit)
  QueryHit で指定された IP Address, Port に direct connection
    gnutella network を経由しない
    (direct TCP connection. not via the gnutella network)

  download protocol is HTTP
    もちろん web サーバではなく servent が受ける
    (received and processed by the servent)

  GET /get/<File Index>/<File Name>/ HTTP/1.0\r\n
  Connection: Keep-Alive\r\n
  Range: bytes=0-\r\n
  User-Agent: Gnutella\r\n
  \r\n

  返事 (reply)

  HTTP 200 OK\r\n
  Server: Gnutella\r\n
  Content-type: application/binary\r\n
  Content-length: 4356789\r\n
  \r\n
  ファイルの内容 (content of the file)

  Range parameter があるので, 途中からのダウンロードも可能
  (partial downloading is possible (after premature disconnection))

Firewalled Servents
  ファイルを保持する servent が NAT の内部にあったら?
  (when the offering servent is in a NAT network)
  前項の direct connection は張れない
  (no direct TCP connection is possible from the outside of the NAT)

  NAT の内側のホストは, 外へは connection 可能
  外向き(つまり逆向き)に connection を張ってもらおう
    ファイル保持 servent → ファイル要求 servent
  (outbound connection is still possible from the offering servent)

  それを依頼するのが Push descriptor
  (Push descriptor tells the offering servent to open connection)

  direct connection の後, 保持 servent → 要求 servent へ
  (after the connection established,)

  GIV <File Index>:<Servent Identifier>/<File Name>\n\n
  (this is not an HTTP request)

  これを受けた要求 servent の動作は前述の通り
  (the requesing servent responds to that as follows)

  GET /get/<File Index>/<File Name>/ HTTP/1.0\r\n
  (以下同様) same as the sequence in normal transfer
-------------------------------------------------------------
Push descriptor
  QueryHit を得て, ファイルの転送接続を要求する
  (receiving QueryHit, requests the offering servent to open connection)
    QueryHit の来た経路を逆にたどって転送
    (routed along the route of QueryHit reversely)

  payload  26 byte
    Servent Identifier (16)
    File Index (4)
    IP Address (4)
    Port (2)
```

  Servent Identifier (contained in the QueryHit message)
    QueryHit の payload に含まれていたもの
    接続要求の宛先ホストを指定
    また, これを使って descriptor の転送経路を知る
    (used to route Push descriptor)

  File Index
    QueryHit の result set にあったファイルのどれかの番号
    (file index contained in the result set of the QueryHit descriptor)

  IP Address, Port
    要求側の servent の受信アドレス
    (of the requesting servent)

---
descriptor routing

TTLを減らす (Hop を増やす)
  (decrease the TTL (and increase the Hop))
  ゼロになったら転送しない
  (descard the descriptor if the TTL reaches 0)
  Hop の値によって破棄してもよい
  (may use Hop value instead of TTL)

重複の防止 (prevent duplication of descriptor)
  同一の (Descriptor ID, Payload Descriptor)をもつdescriptor
    できるだけ転送しないように
  (identity of the descriptor is (Descriptor ID, Payload Descriptor))
  discard duplicated messages
    using message history

Ping, Query
  それを受信した以外のすべての接続に転送 (flooding)
  (route the descriptor to all connections (except the incomming one))

Pong
  対応するPing (Descriptor ID が同じ) が来た接続だけに転送
  (route the descriptor to the connection from which the corresponding
   Ping has arrived before)
    対応するPingがなければ, 破棄
    (discard if no correspoinding Ping exist)

QueryHit
  対応するQuery (Descriptor ID が同じ) が来た接続だけに転送
  (route the descriptor to the connection from which the corresponding
   Query has arrived before)
    対応するQueryがなければ, 破棄
    (discard if no correspoinding Query exist)

Push
  対応するQueryHit (Servent ID が同じ) が来た接続だけに転送
  (route the descriptor to the connection from which the corresponding
   QueryHit has arrived before - with the same Servent ID)
    対応するQueryHitがなければ, 破棄
    (discard if no correspoinding QueryHit exist)

Freenet
"Freenet: A Distributed Anonymous Information Storage and Retrieval System"
(1999)

design goals:
  Anonymity for both producers and consumers of information
  Deniability for storers of information
  Resistance to attempts by third parties to deny access to information
  Efficient dynamic storage and routing of information
  Decentralization of all network functions

本来の目的
  言論の自由 (freedom of speech)
  匿名性 (anonimity)
    著作者 (for producers)
    閲覧者 (for consumers)
    保持者 (for data storage owners)
  削除, 改竄への対抗 (anti-attacks)

本講義での着目点 (however, in this lecture ...)
  P2Pにおけるファイル保持システムとして (technical aspect as a p2p system)
  Unstructured と Structured の中間
    gnutella のような flooding にはたよらない (no flooding)
    でも, 探索(バックトラック)がある (with backtrackings)
    必ず見つかるとは限らない (no guarantee of search success)
---
3. Architecture

Routing table
  host(IP/port) + keys

ディスクの領域を提供 (offer data storage)

ファイルはキーで識別 (file is identified by a key)
  160 bit

ファイル検索とは, キーからファイルを入手するプロセス
(searching is the process of matching keys to files)

routing table
  (ノード, {そのノードから得られたファイルのキー})
        (node, {key of the file retrieved from the node})

3.2 Retrieving data

the user generate a query message, and hand it to the
local node

リクエストが来る (エンドユーザから, 他ノードから)
  (on arriving requests,)
  検索すべきキー (contains key to search)

ローカルに保持していれば (if the node hold the key (and the file))
  それを返す (ファイル本体 + 保持しているノード情報) (return that)
                              =自分

ローカルになければ, routing table を用いて転送
  (or, forward the request)
  最も近いキーを得たことがあるノードへ
  (to the node with the nearest key)
    近い = 辞書順 ('near' means lexicographically)
  成功 (if got success message)
    それを返す (ファイル本体 + 保持しているノード情報) (return that)
    routing table を更新 (update local routing table)
      (保持しているノード情報, キー)を追加登録 (add the success information)
  失敗 (if got failure message)
    次に近いキーを得たことがあるノードへ (backtrack)
  隣接ノードの候補がなくなれば (if there is no node to forward)
    失敗を返す (return failure)

  steepest-ascent hill-climbing search with backtrack

  Figure 1

TTLメカニズム - hops-to-live

ループの検出 (loop detection)
  「見た」メッセージを記録 (record the ID of seen messages)

データのキャシュ (caching of the file)
  ファイルを要求元に転送する際に, 自分でも保存する (when the node relay it)

成功の返事の修正 (modify the 'owner' field of success message)
  返事の「保持しているノード情報」を変更してよい
  たとえば自分にするとか (e.g. to the address of the node)
    情報の origin を隠す (in order to hide the real owner)
    request の集中を防ぐ?

利点 (merit of the method)
  成功した query が routing table に反映される
      (succeeded query modifies the routing tables)
  その結果として (as a result,)
  前回検索に成功したキーに近いキー
  (keys similar to the key succeded are)
    その方向(ノードdとする)に転送される
    (forwarded to the 'right' node d)
    ノードdは(今回は知らないので)さらに転送して検索
      (d forwards it to the correct node eventually)
    発見できればそれを返す

  二つの観点から, システムが進化していく
  (the system evolves in two aspects)

  ○ ノードdはそのキーに近いqueryを多数受ける
     (d receives many queries similar to the key originally answered)
    ますますその方面の知識が充実
    (d will have much knowledge about similar keys)

  ○ 転送において, 途中のノードがファイルをキャシュする
    (caching of the file in intermediate nodes)
    知識の充実だけでなく, ファイルも集中する
    (d will have also similar files)
    (近いキーをもつ)ファイルのクラスタ化
    (resulting the clustering of files with similar keys)

3.3 Storing data
  request とほぼ同様
  まずキーを生成して最寄りのノードへ送信
  (first, generate a key for the file and send it to a neighbor node)
    TTL を指定する
      (データを格納するノード数を指定したことになる)
      (TTL works as the number of nodes to store the file)
  受信したノードは (receiving store message,)
    すでにローカルにあるかチェック (check if the node has the file already)
    あれば request と同様に既存のファイルを返送 (if it has, return the file)
      ユーザは衝突を知る (the inserter knows the collision)
      キーを変更するとかの対策をとる (she may change the key and try again)
    なければ隣接ノードから最適なノードを選び転送
    (if not, forward to the node with the nearest key from the routing table)
      (=キーが近い)
    TTLが尽きた - うまくいった (TTL expired -- it is the success of insertion)
      all clear が返る

  次にユーザはファイル本体を送信 (next, send the file)
    さっきの経路で進む (proceed along the way as in the last step)
    途中のノードは (intermediate nodes caches the file,)
      ファイルを保存
      routing table更新 (and update the routing table)
        (inserter, key)を登録

  経路のループなどで失敗したら (if failed by e.g. loops,)
    request 同様 backtrack

利点 (merit of the method)

「格納の際, あとで探すときに探しそうな場所に置く」
(store the file to the node which will be asked afterwards)
hashtableの格納アルゴリズムとの類似 (similarity to the hashtable)

コピー数をTTLで制御 (control the number of copies by the value of TTL)

同じキーを持つ偽データで上書きさせようとすると
(if malicious user try to overwrite a file with the same key,)
逆に本物(先客)がさらにひろがる
(the original file will spread)

Adding nodes
まず初期接続ノードを発見してrouting tableに登録
(find some initial node and add to the routing table)

その時点で既存ファイルの request, insert はすぐにできる(前述)
(at this point, file request and insertion are possible)

ファイルを保持するためには (but, to offer a disk space,)
既存のノードに存在を知ってもらう必要
(existing nodes should know the new node)
既存ノードの routing table にエントリを確保
(i.e. add the new node in the routing table of existing nodes)
その新規ノードが担当するキーを定める必要
(for that, the new node need a key)

そのキーは
みんな（キーを持つ人）が合意する必要
(the key should be consistent in the routing table of other nodes)
でも誰かが勝手に決められては危険
(the new node should not determine the key arbitrally)
任意のキーを担当するノードが作られるとよくない

グループで乱数を作るプロトコルが存在
(a protocol to generate a random key by a group of nodes)
参加メッセージをランダムに転送しつつ
(along the forwarding of the entry message,)
各ノードが乱数を発生
(each node generates a random number)
TTL が尽きたところで, 戻りつつ??各ノードの乱数を XOR
それを新規ノードのキーとする
(on the expiration of the entry message, XOR of the all random numbers
will be the key of the new node)

3.1 検索キーのいろいろ (search keys)
ハッシュの利用 (the hash algorithm)
SHA-1 (Secure Hash Algorithm)
160 bit のハッシュ値を生成
3種類のキーが存在
KSK (Keyword-signed key)
SSK (signed-subspace key)
CHK (content-hash key)

KSK - ファイルの名前(など)から検索キーを作る
(generate a key from the name of the file)
まず, descriptive text を作る
(説明文 あるいは ファイルの名前)
その descriptive text から, public/private key pair 生成
この生成は deterministic - 再現性がある
public key をハッシュして, 検索キーとする
(the search key is obtained by hashing the public key)
ファイル本体には private key で署名(sign)

ファイルを公開(publish)するには, descriptive text を公開する
それを手に入れた人は, 上述の手順を行なう
(once the text is obtained, users follow the process explained above)
検索キーを生成して, ファイルを入手
(generate the search key and retrieve the file)

ファイル本体の署名も確認できる - 改竄されていない
(the file may be checked against the private key)

descriptive text の名前空間がひとつしかないのが問題点
(KSK is in a single namespace)
衝突 (collision)

SSK - 著作者ごとの名前空間を導入 (enables personal namespaces)
著作者は public/private key pair を秘密に作成
descriptive text と public key から, 検索キーを生成
ファイル本体は private key で署名(sign)

ファイルを公開するには,
(public key, descriptive text)を公開 (publish)

ファイルを入手するには, これらから検索キーを生成して検索
(users generate the search key from these information)
public key でファイル本体の署名を確認

著作者以外は private key がないので,
(only the author has the private key,)
新しいファイルを公開できない
(other user cannot insert files)

CHK - ファイル内容のハッシュによる検索キー
(hash result of the content of the file)

ファイルを公開するには, 検索キーがあれば十分
