Freenet, Gnutella, Napster

Currently Popular Peer-to-Peer Systems

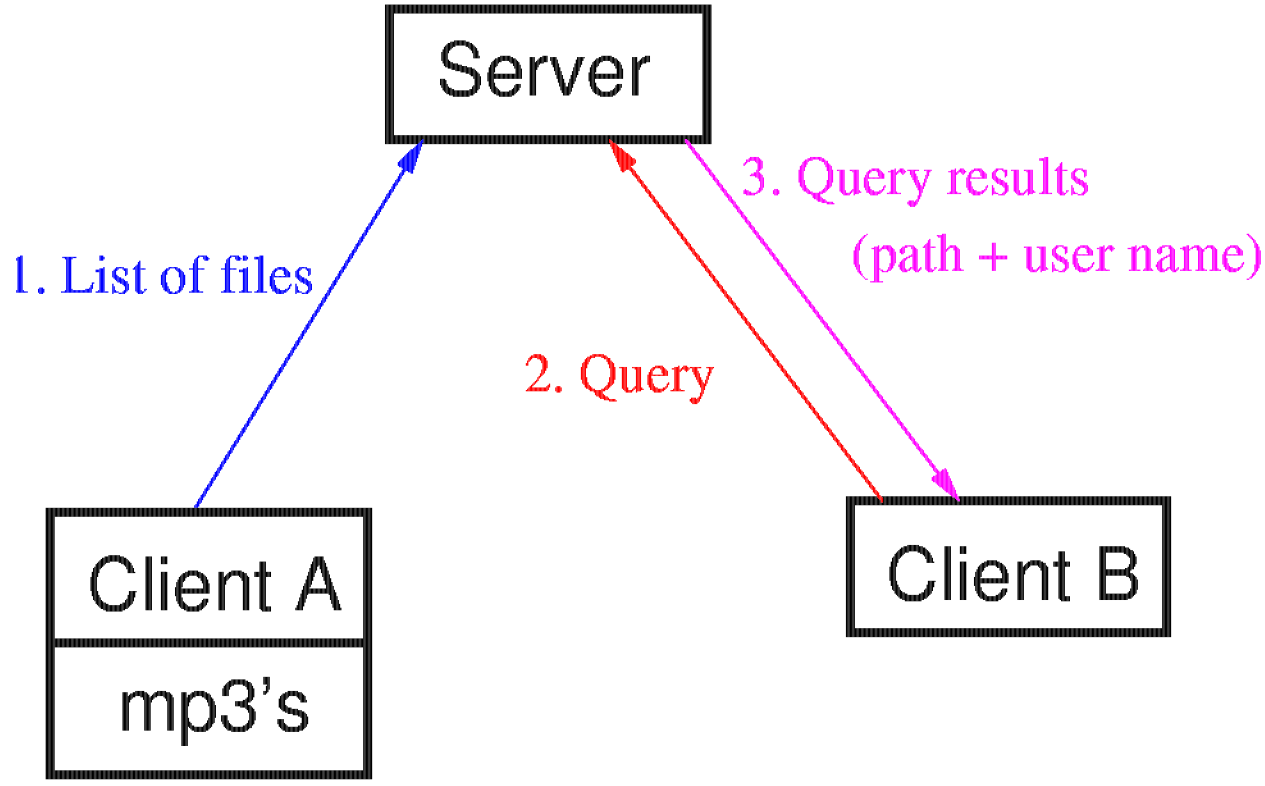
Dipl. Inform. Arno Wagner
Communication Systems Group, ETH Zürich

Example I: Napster

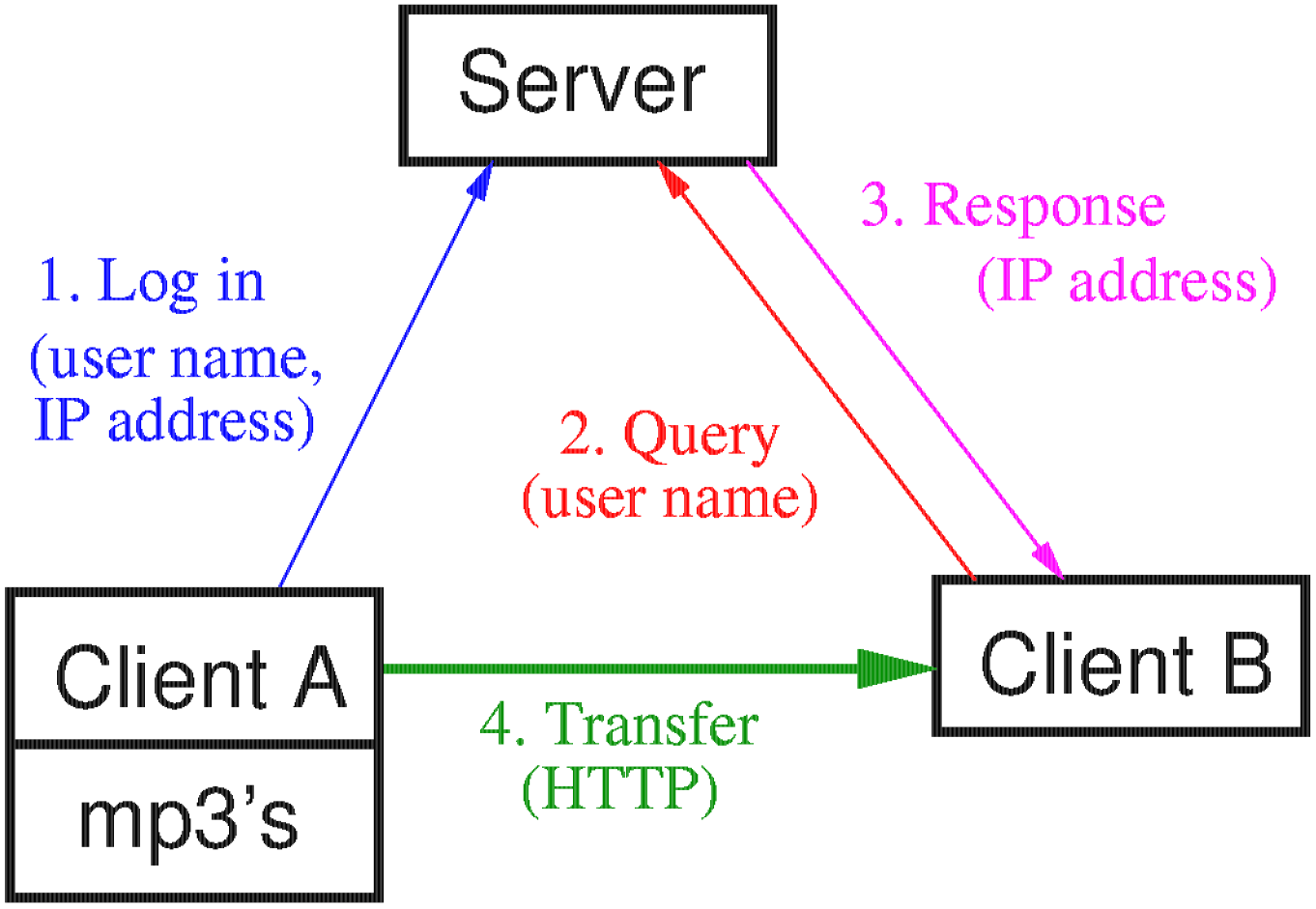
- Commercial product of Napster, Inc.
  Free alternatives ⇒ http://opennap.sourceforge.net
- List of (free) servers under http://www.napigator.com/list.php
- **Purpose:**
  - Sharing of MP3 files users have on their local storage.
  - Search in central database.
  - "Current" Business Model: License/Subscription fees (?)
- Online since 16.11.1999
- Client-Server architecture
  - Client: MS Windows / numerous clones
  - Server: not available / opennap for Unix

Structure: File Offer and Search

Search type: substring



File Transfer



Anonymity

- **Server**
  - Knows user name, IP-address and files offered
  - Knows file transfer *attempts* (not transferred files)
- **Client**
  - Offerer
    - Does not know about searches
    - File transfer: knows IP-address of taker
  - Taker
    - Search: Only user name and path (file name) visible
    - File transfer: knows IP address of offerer

Anonymity: Limited towards other participants, none towards server

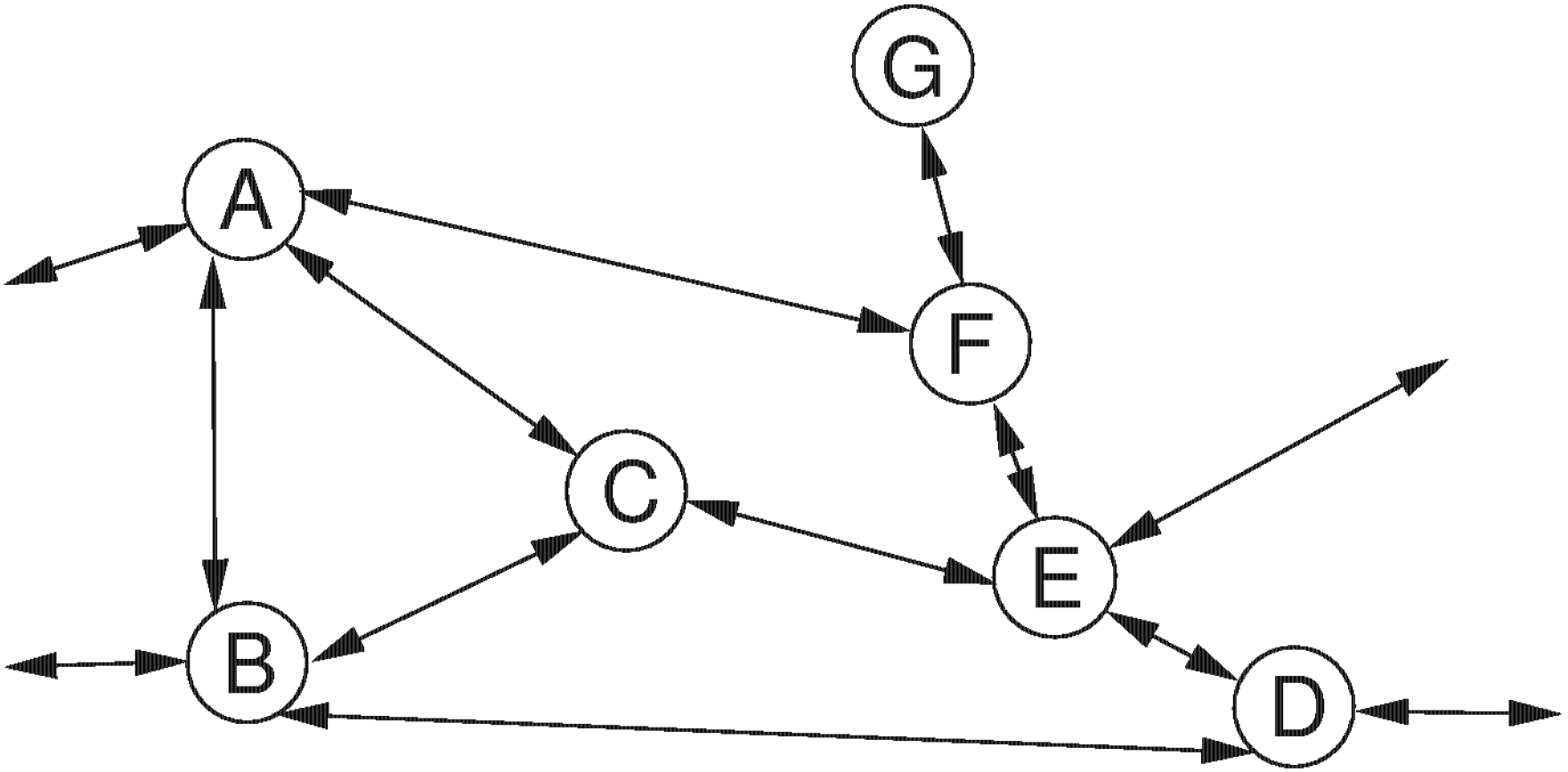
Summary Napster

- Commercial foundation, by now non-commercial alternatives
- Centralistic architecture
- Technical and legal attacks on server are easy:
  - March 2001: Court ordered filtering $\Rightarrow$ 60% traffic drop
  - May 2001: Filtering problems due to alternate naming schemes
  - Legal attacks on opennap servers
  - Legal/technical attacks on users (only MS client)
- Bottleneck: Server bandwidth

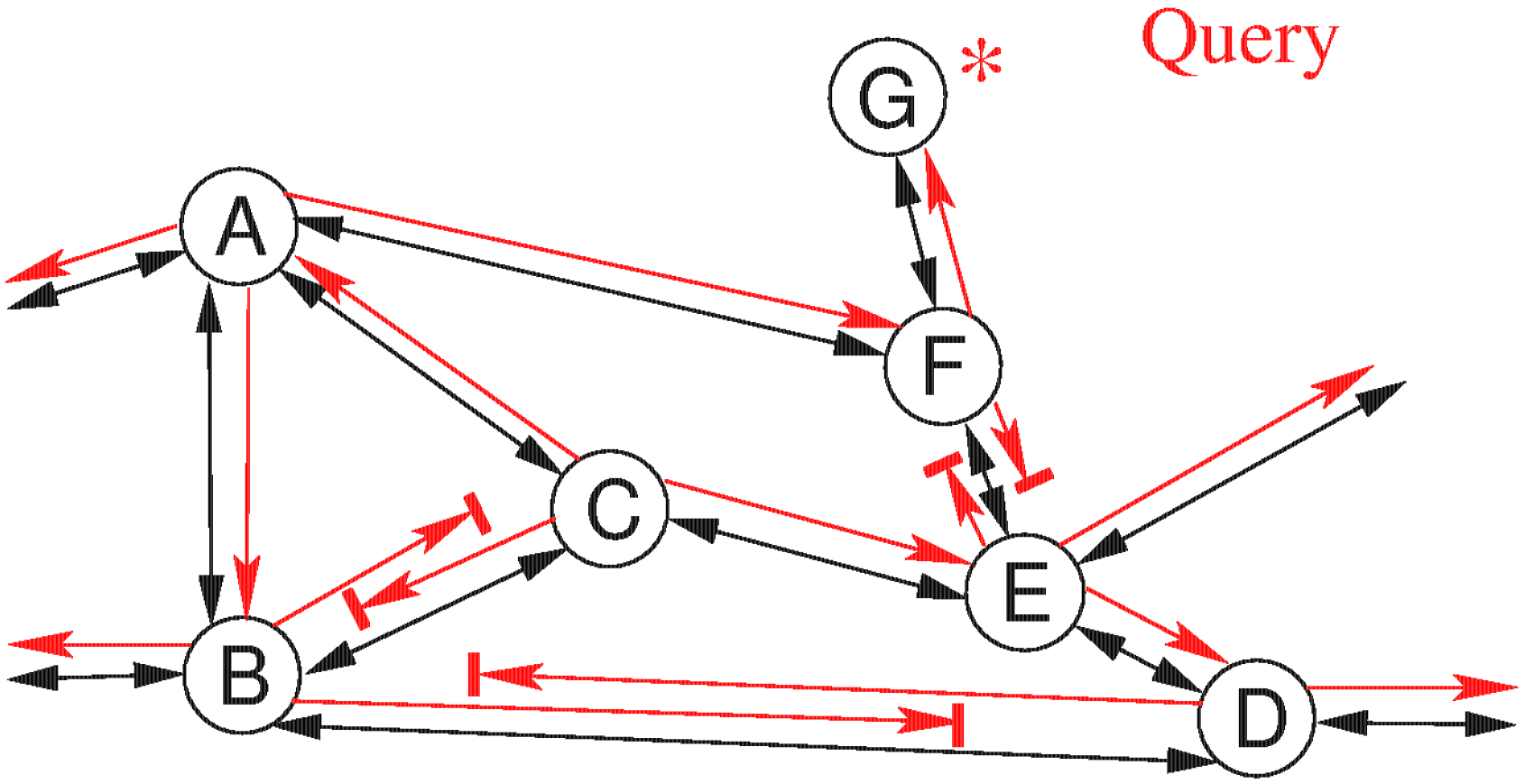
Example II: Gnutella

- "Successor" of Napster, GPL'ed
- **Characteristics**
  - Sharing of arbitrary files
  - Distributed searching
  - No central infastructure
- Publicised by Nullsoft: Sources on the web for some hours on 14.3.2000, but then suppressed by AOL/Time Warner
- Open further development (ressources: http://gnutella.wego.com)
- Client = server ("servent"), "self-organized"
- Servent for most current platforms

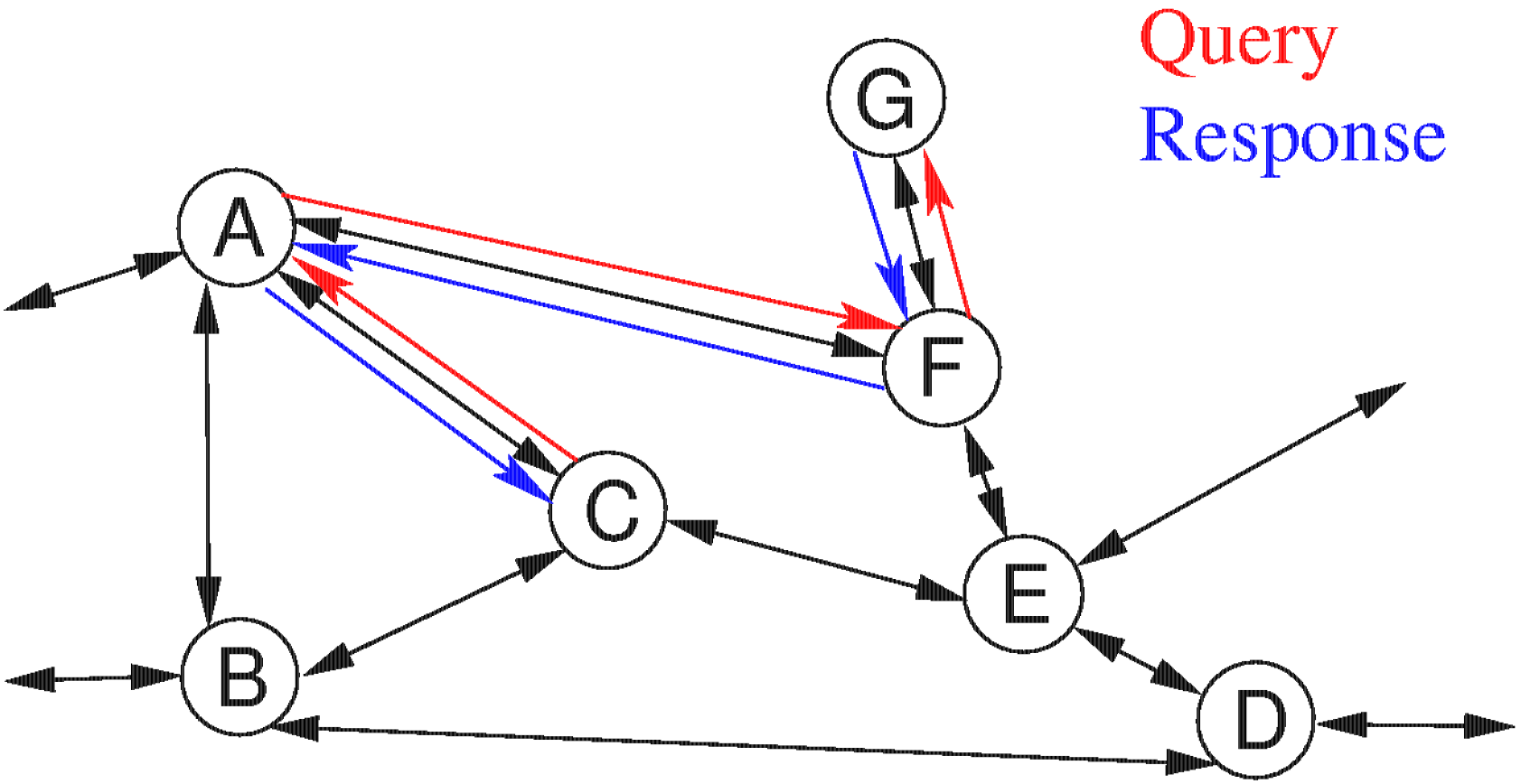
Topology



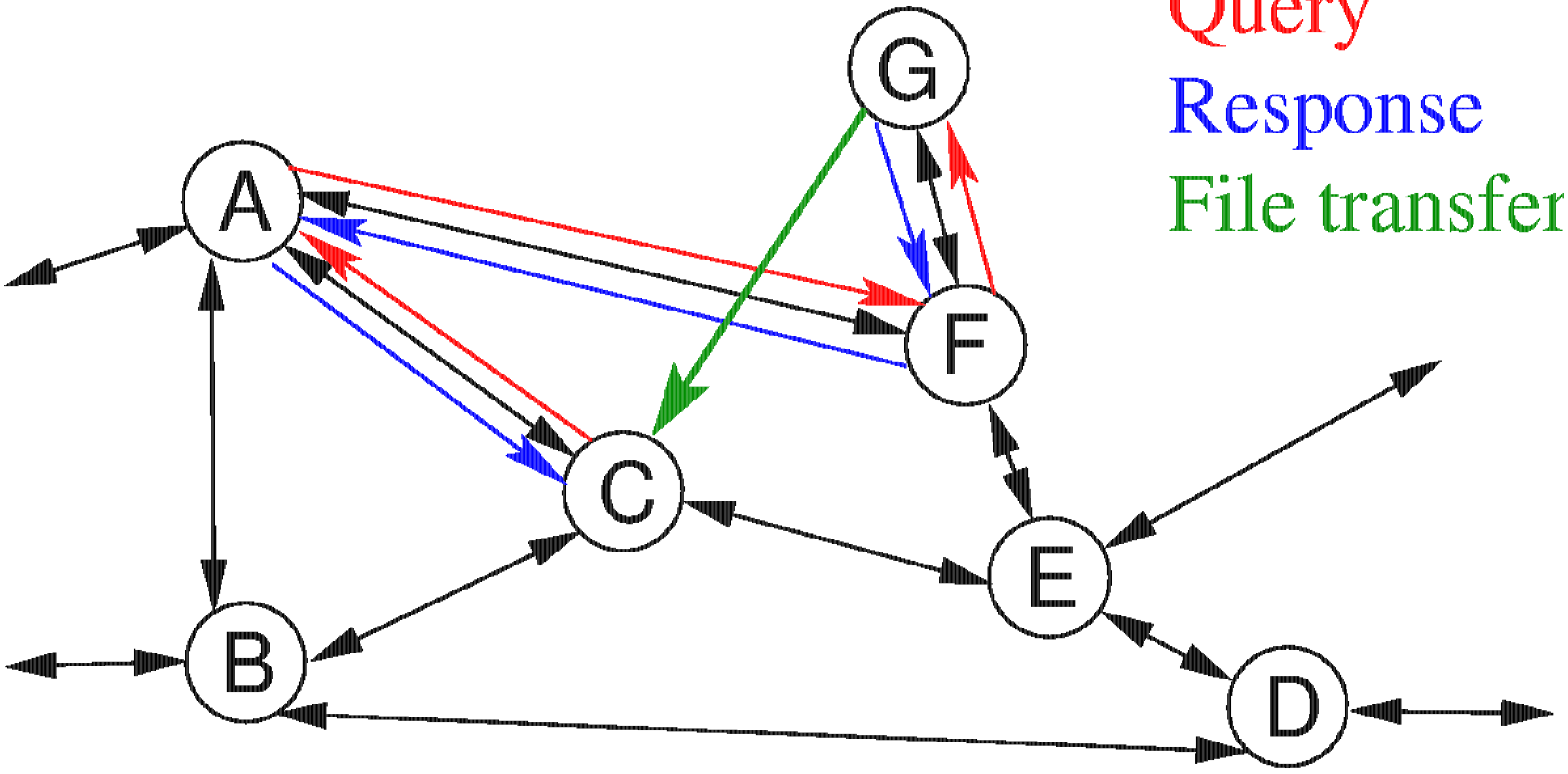
Query from C, hit at G



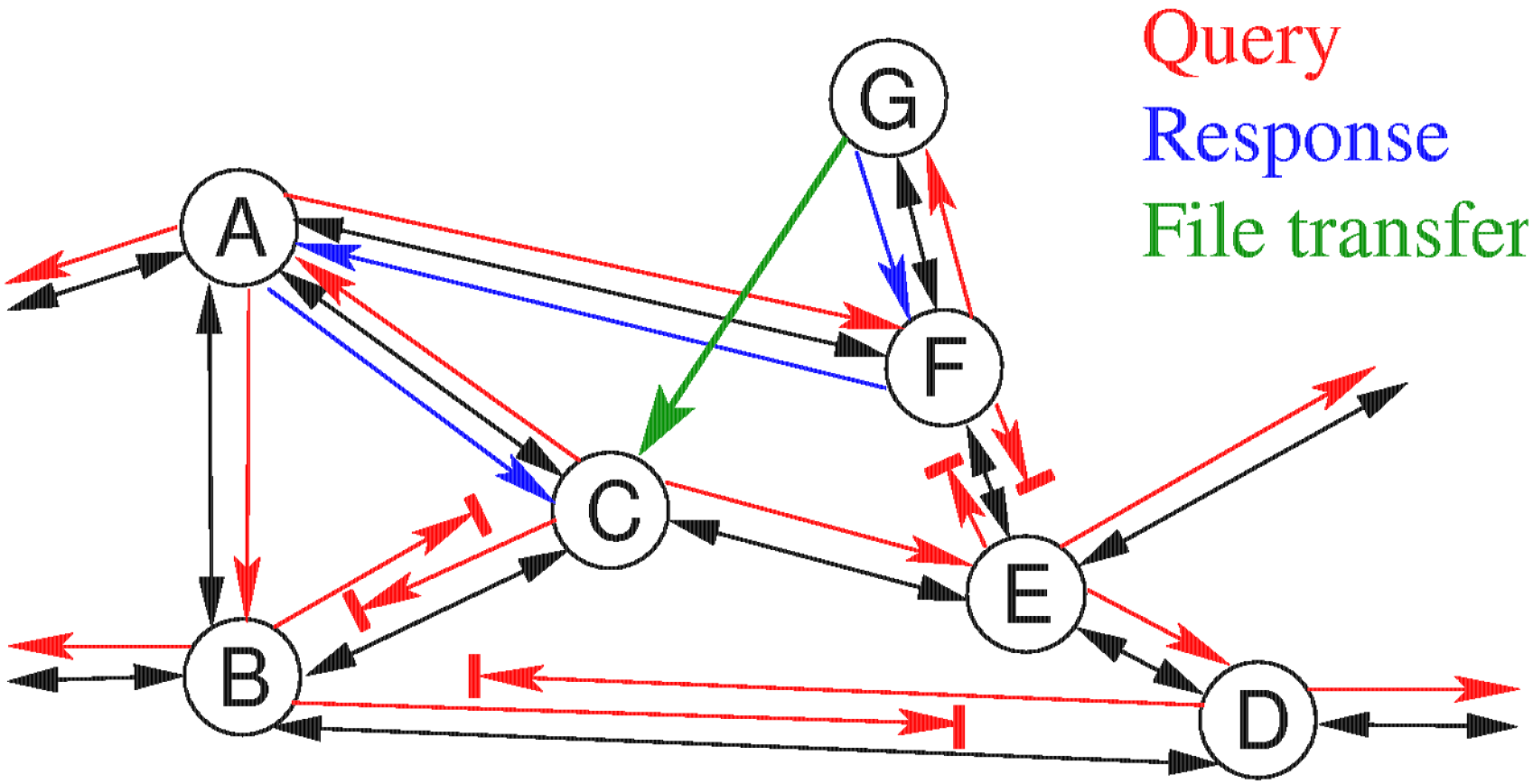
Response from G to C with query result



Filetransfer from G to C



Complete View



Anonymity

- **Queries**
  - Offerer: Does not know query source
  - Query source: Gets IP address of offerer
  - Intermediate node: Gets query and offerer IP address
- **File transfer:**
  Direct contact, everything knowen $\Rightarrow$ Hall of Shame

Remarks / Problems

- Chaotic topology: Might not even be connected!
- Highly dynanmic node set
- Queries via depth limited breadth-first search
  - Slow (up to minutes)
  - Might only reach part of the net
  - High network load
- "Log On" with list of knowen nodes ⇒ problematic!
- Immediate resharing: Many truncated files offered!
- Limited downloads: Found files cannot be transferred, up to 100% failure!
- Limited public service: Only 30% of all nodes offer files.

Historical Development

- June 2000:
  - Network load about ca. 2 kB/sec per connection
  - About 5000 nodes active at any time
- November 2000:
  - 10.000 - 30.000 nodes active
  - Fragmentation into clusters with up to 7000 nodes
  - Up to 1200 responsive nodes in a segment
- May 2001: Comparable to November.
  - Network load about 1kB/sec per connection.

$\Rightarrow$ Statistics at http://dss.clip2.com/gnutella.html

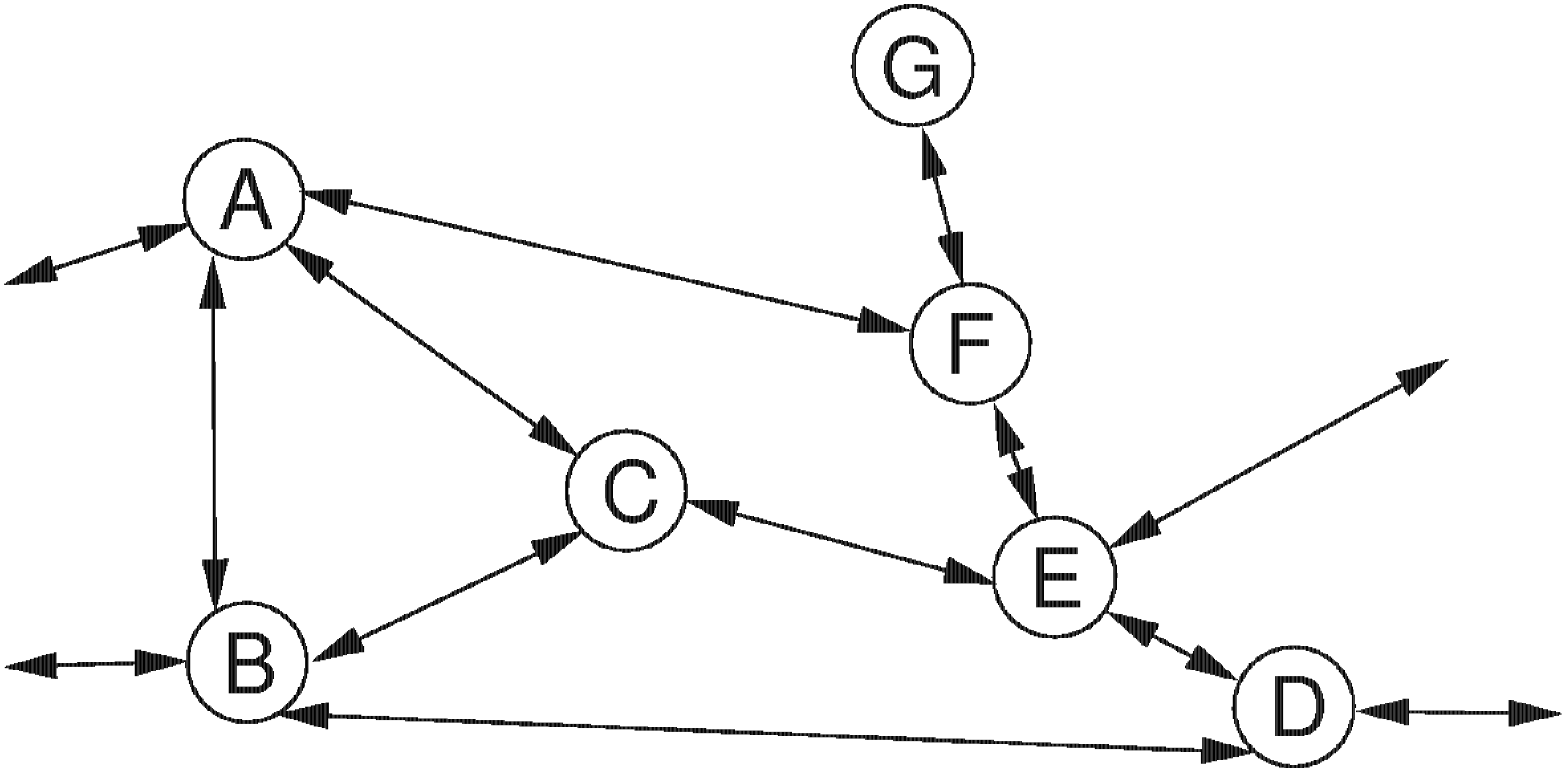
Summary Gnutella

- No central infrastructure (not even DNS)
- Robust against technical attacks
- Limited robustness against legal attacks
- Systematic search difficult/impossible
- Reaction to growth was fragmentation $\Rightarrow$ limited scalability
- The future: Reflectors?

Example III: Freenet

- **Aim:** Ensure free speech $\Rightarrow$ "WWW" without the possibility to censor
- **Foundation:** Master Thesis of Ian Clarke
- Project homepage: http://freenetproject.org
- Client = server, self-organizing
- Development in JAVA
- **Status:** Work in progress, limited functionality (no search)

Topology



Topology

Like Gnutella, but

- Data is also routed
- Data (and direct origin) cached in intermediate nodes
- Data does only exist in caches
- Cache freeing strategy:
  1. Early deletion of data
  2. Late deletion of data origin $\Rightarrow$ Fast results for repeated query

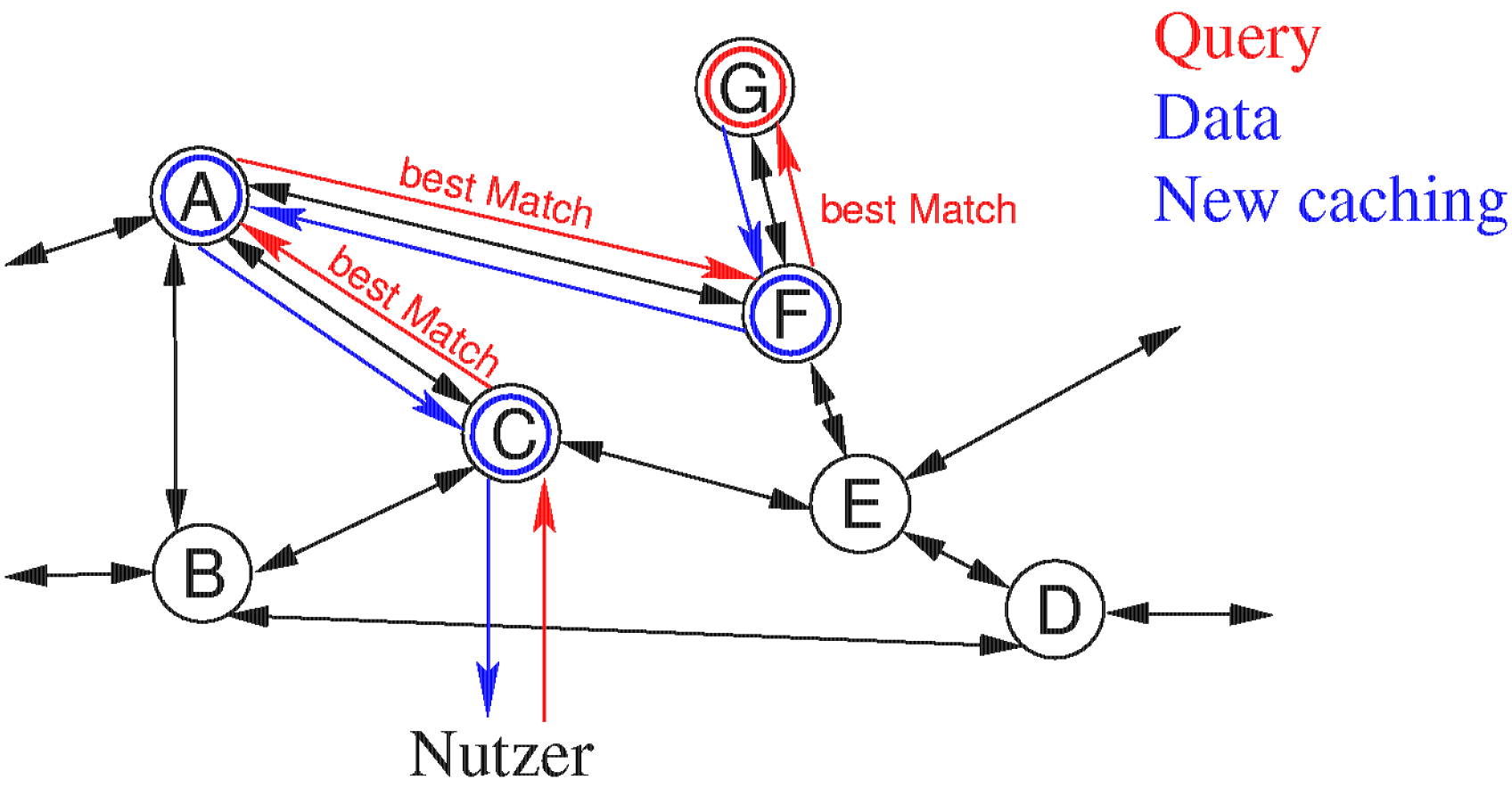
Search and Response

- String search ("keys" by crypto-hash)
- Data with similar keys form "clusters".
- Query response is the found data
- Search strategy:
  1. Local search for similar keys in cache
  2. Most similar $\neq$ search target:
     Continue where most similar was obtained from

  $\Rightarrow$ self-organization, data-load balancing

Problem: How to know what to search for?
Current "solution": Publication of search strings

Search example



Anonymity

- Overall good:
  - Full routing, document source cannot be identified
  - No master-copies of documents exist
- No plain text documents $\Rightarrow$ node administrator has no knowledge of data at node
  Present day: Search string lists problematic

Summary Freenet

- High anonymity
- Scalability not known
- Searching not solved
- Data throughput low, not usable for large files
- Expected search times unclear (today 1-60 sec)
- Data clustering working in large networks?
- Absence of deletion possibility really wanted?