Peer-to-Peer Networks

Practical Exercise 5

Dominik Fischer

P2P Networks Group at TU Darmstadt

22. January 2013

Kademlia Overlay Evaluation

- everybody was on time and achieved full point value
- each and every time I hope to keep that line
- but then the attestations take place. . .

Native Multicast

- message sent by one socket,
- received by many
- introduces as little overhead as possible
- by replication packages on demand in routers
- utilizes IP addresses with in high values
- fixes the implementation to an address type

Implementing Multicast

- C programmers depend on their `sockopts`.
  - Tack and Sean Walton provide tutorials.
- Python largely recycles the C methods
  - Doug Hellmann has some words about and
  - it was discussed on Stack Overflow.
- Java's `UDPChannel` provides a `MulticastChannel` interface.
  - The NIO2 primer has an example on slide 14 ff.
- .NET also provides uniform interface methods.
  - A Tutorial by Kelly Elias provides a good start.

Using Multicast

- peers can communicate anonymously without knowing
  - who listens
  - if anyone listens at all
- multicast commands and queries to various peers
- peers join groups according to given criteria
  - peers having data
  - peers having files
  - unconnected peers
  - erroneous peers
- and can be controlled at once altogether

Programming Exercise 5

- Make an educated guess. . .
- complete Kademlia
- storing key-value-pairs
- retrieving them
- `FIND_VALUE`, `STORE`
- data storage in RAM is sufficient
- regular timeout and re-publishing of entries
- cache when found

Task Considerations

- files are dissected into parts
  - choose a reasonable size
  - choose a reasonable hash or identity function
  - smaller files are stored immediate

storing files exemplified

$$\#(name(file)) \rightarrow \left\{ \begin{array}{ccc} \#(part_1) & \rightarrow & part_1 \\ \#(part_2) & \rightarrow & part_2 \\ \vdots & & \vdots \\ \#(part_n) & \rightarrow & part_n \end{array} \right\} content(file)$$

- demonstrate split and reassembly of files

Schedule

- 20 points to achieve
- 3 weeks time, last programming task
  - 29. January - theory
  - 5. February - no class
  - 12. February - questions and answers