The GNUnet DHT

Christian Grothoff
christian@grothoff.org
http://grothoff.org/christian/

"I trust no one, not even myself." –Joseph Stalin

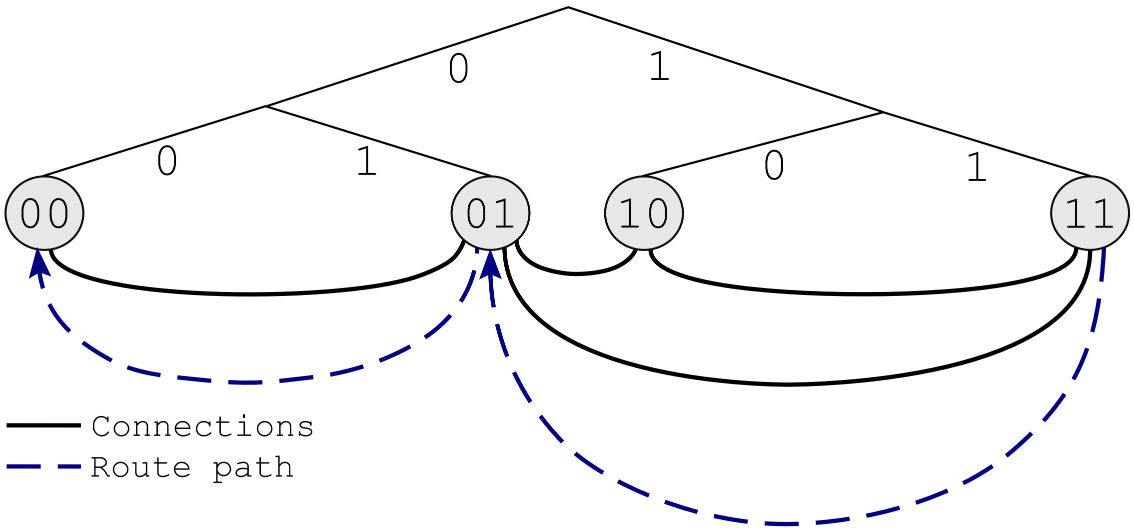
Agenda

- A Quick Introduction to Bloom Filters
- The R^5N Routing Algorithm
- Performance Analysis for R^5N
- Content validation
- The DHT API
- The BLOCK API

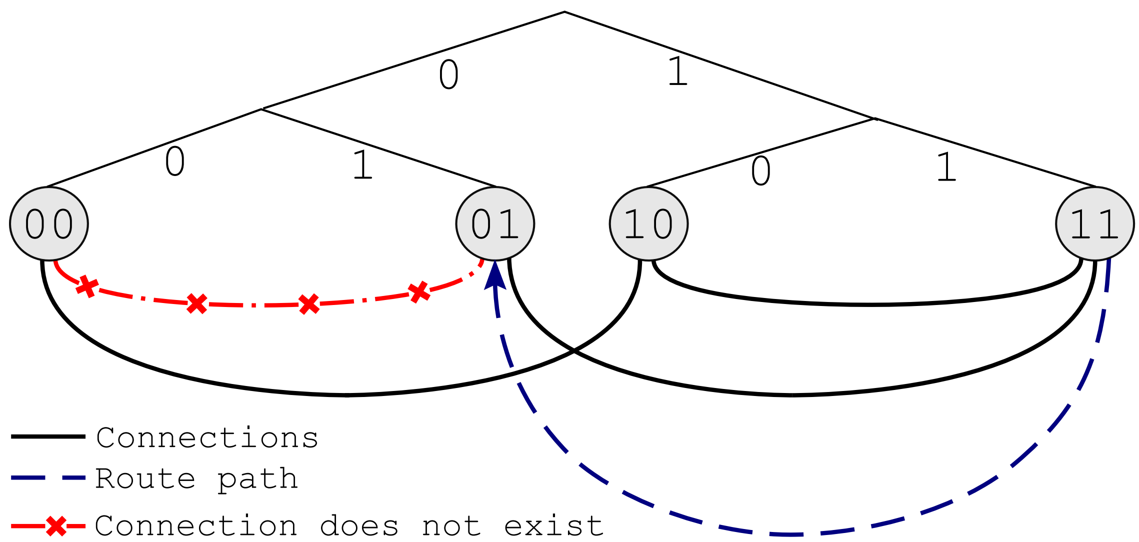
Bloom Filters

- Probabilistic data structure to answer the question "is element X in set S" with "no" or "maybe"
- If an element is not in the set, the probability is high that the answer is "no"
- Uses a bit-array where k bits based on $H(X)$ are set to 1 for each element $X \in S$.

Review: Kademlia



Kademlia and Restricted Routes



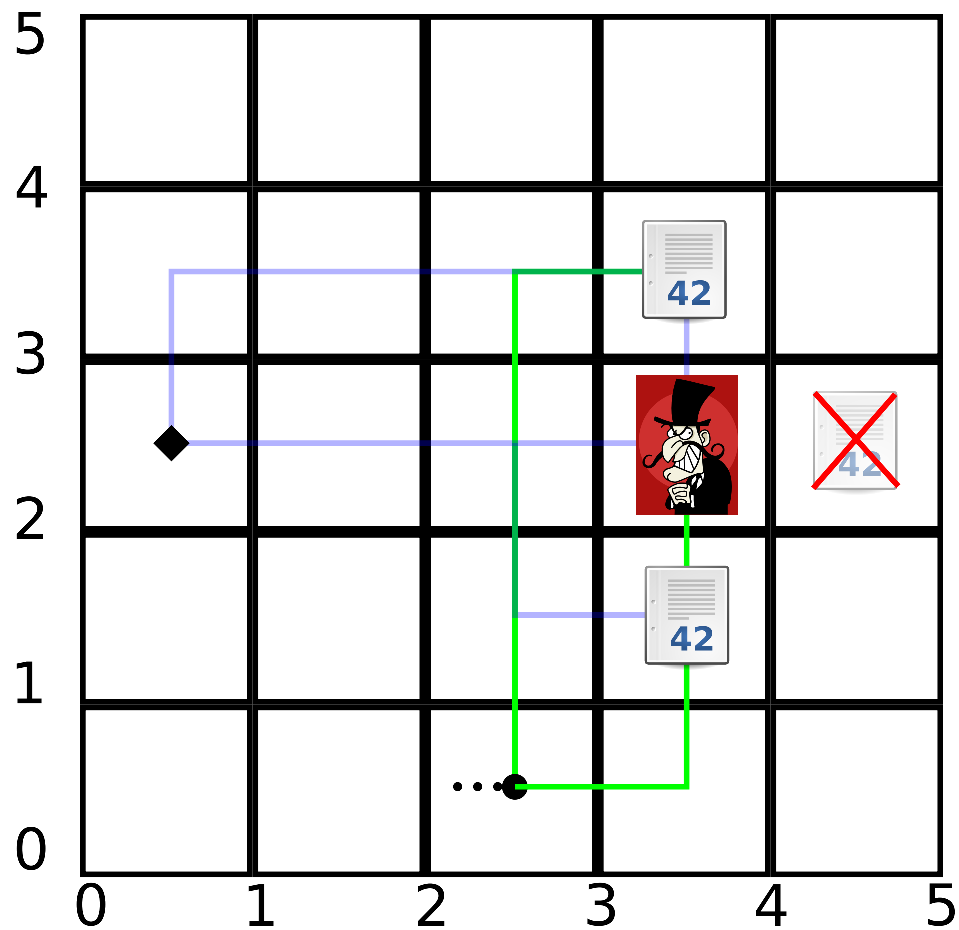
The R^5N Routing Algorithm

- Designed to work well in restricted route networks (many nearest peers) and reduce the impact of malicious peers.
- Requires recursive routing; less control for initiator, better performance; stateful return routing
- Kademlia style routing table — so-called "k-buckets" storing k peers; such that the i^{th} k-bucket stores peers with XOR distance between $[2^i, 2^{i+1})$

The R^5N Routing Algorithm

- Random and Kademlia style routing phases
  $\Rightarrow$ combines path *diversity* with *efficient* routing
  - Random phase: "start" Kademlia routing from random location.
  - Kademlia phase: efficiently find nearest peers.
- Requests have desired replication level r; the number of nearest peers a request *should* reach.
- Achieved by probabilistic path branching, at each hop a request may be forwarded to one or more peers.

The R^5N Routing Algorithm



The R^5N Routing Algorithm

PUT Request

if $nearest(r)$ **then**
  $store\_data(r)$
else
  for $i = 0 \rightarrow num\_forwards(r)$ **do**
    $p = get\_forward\_peer(r)$
    $forward\_request(r, p)$
  end for
end if

GET Request

if $NULL \neq (d = find\_data(r))$ **then**
  $route\_result(r, d)$
end if
for $i = 0 \rightarrow num\_forwards(r)$ **do**
  $p = get\_forward\_peer(r)$
  $store\_route(p, r)$
  $forward\_request(r, p)$
end for

Routes with Loops

- R^5N cannot loop forever due to the hop counter
- Looping is still inefficient

$\Rightarrow$ R^5N uses a Bloom filter to avoid loops

Performance Analysis for R^5N

- Randomized routing takes c steps, $c \sim \log n$
- Kademlia-style routing takes $O(\log n)$ steps

$\Rightarrow$ Finding a nearest peer is $O(\log n)$

Performance Analysis for R^5N

- There are $\frac{|N|^2}{|E|} \in O(|N|)$ nearest peers
- For a 50% success rate for a single GET, we need $O(\sqrt{|N|})$ replicas
- Then repeat GET $O(\sqrt{|N|})$ times for "high" success rate

$\Rightarrow$ Total routing cost is $O(\sqrt{n} \log n)$

Absolute Performance

Size of network	Average hops per PUT		Average hops per GET	
	R-Kademlia	R^5N	R-Kademlia	R^5N
100	2.70 ± 0.06	3.96 ± 0.06	2.54 ± 0.03	4.63 ± 0.17
250	3.06 ± 0.10	4.26 ± 0.10	3.10 ± 0.06	5.96 ± 0.27
500	3.08 ± 0.46	4.38 ± 0.45	3.38 ± 0.06	6.17 ± 1.14
750	3.19 ± 0.74	4.37 ± 0.83	3.50 ± 0.04	6.29 ± 1.04
1000	3.63 ± 0.07	4.47 ± 0.93	3.64 ± 0.04	7.29 ± 0.95

The DHT API

- GNUNET_DHT_connect, GNUNET_DHT_disconnect
- GNUNET_DHT_put, GNUNET_DHT_put_cancel
- GNUNET_DHT_get_start, GNUNET_DHT_get_stop
- GNUNET_DHT_monitor_start, GNUNET_DHT_monitor_stop

Special GET Options

GET requests can be given the following optional options:

- Bloom Filter: filter known results (duplicates)
- Bloom Filter Mutator: change hash function of Bloom Filter
- eXtended Query: additional query information beyond the hash

Options for GET and PUT

- GNUNET_DHT_RO_DEMULTIPLEX_EVERYWHERE
- GNUNET_DHT_RO_RECORD_ROUTE
- Replication level
- Expiration time (provided to PUT, returned by GET)
- Block type $\Rightarrow$ for content validation

Monitoring

DHT monitoring is useful for...

- Testing / debugging
- Performance analysis
- Application development!

The BLOCK API

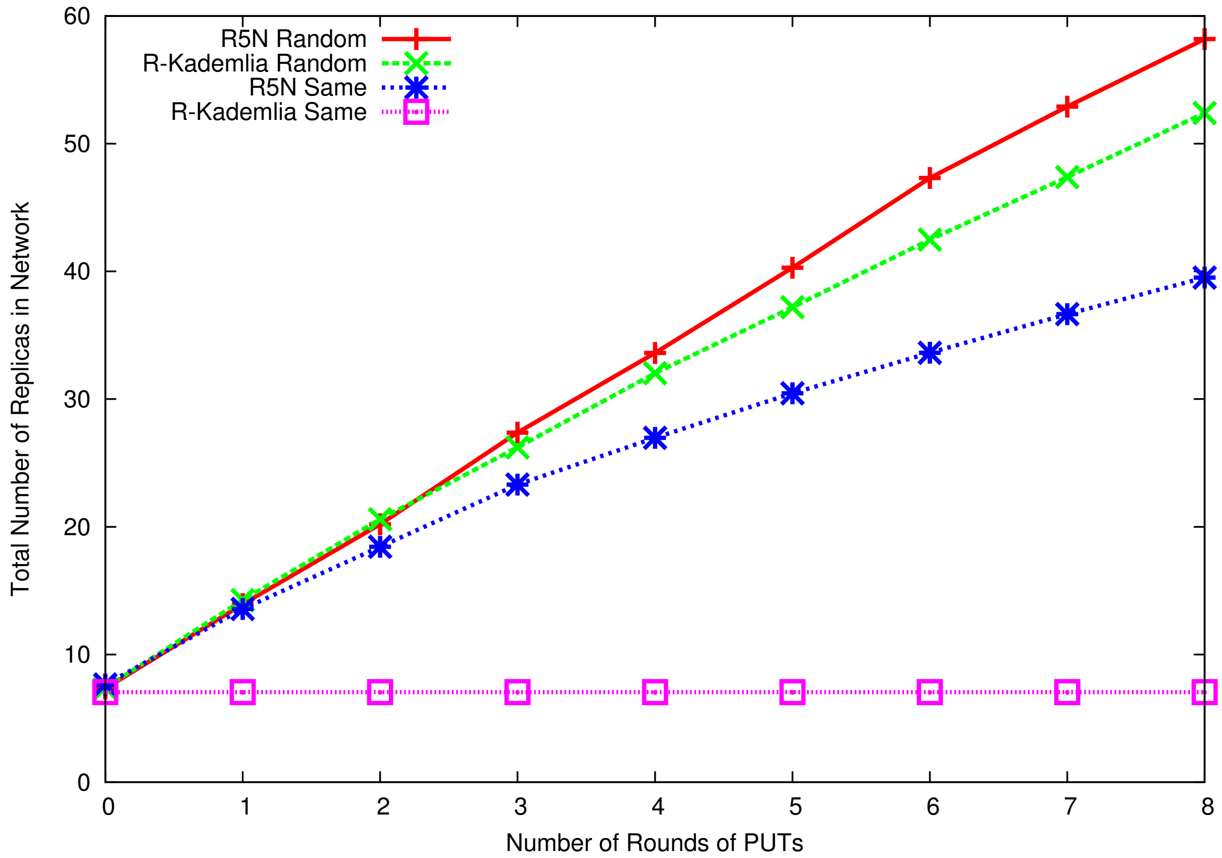
- Block type determines responsible Block plugin
- Configuration option `[block] PLUGINS` specifies supported plugins
- Implement a new plugin based on the `gnunet_block_plugin.h` header
- "fs" for file-sharing, "dht" for DHT internals, "test" for no verification (any data can match any key)

The BLOCK Plugin API

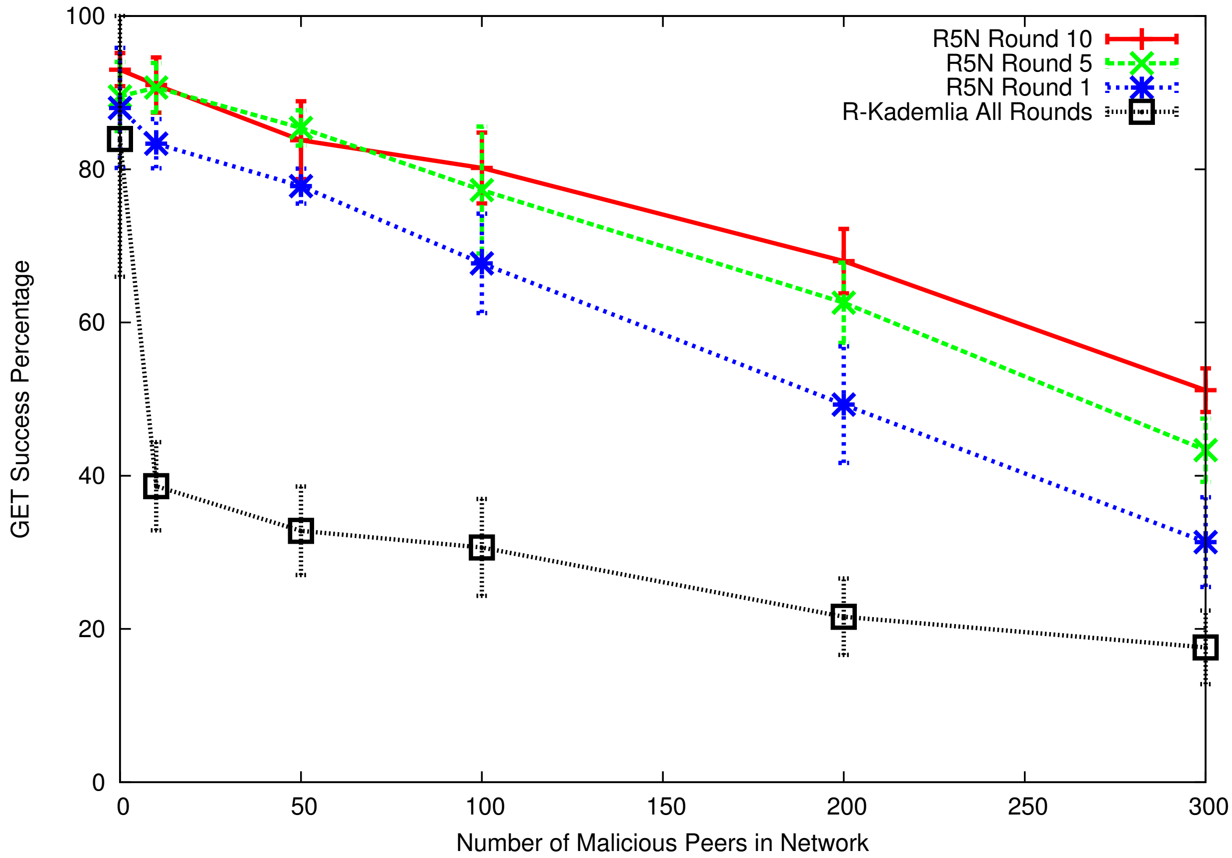
Each plugin must provide two functions:

- `GNUNET_BLOCK_EvaluationFunction`: does the given block satisfy the requirements of the given query? Possible answers include: Yes, and other replies can exist; yes, and this is the only answer; no, duplicate reply; no, invalid reply
- `GNUNET_BLOCK_GetKeyFunction`: given a block, what key should it be stored under? Possible answers are: A key; bad block; not supported

Experimental Results: Replication



Experimental Results: Sybils



DHTs and your Projects

- Distributed Search:
  - Distributing sites to spider?
  - Distributing search queries & replies?
- Jugendschutz:
  - Finding rankings?
  - Finding culturally close users?
- Twitter Censorship:
  - Finding peers monitoring similar feeds?
  - Finding missing tweets?

Copyright