The R^5N Distributed Hash Table

I-D: `https://datatracker.ietf.org/doc/draft-schanzen-r5n/`

IETF118

Martin Schanzenbach ♠, Christian Grothoff ♣, Bernd Fix ♢

06/11/2023

♠ Fraunhofer AISEC `https://aisec.fraunhofer.de`
♣ Berner Fachhochschule `https://bfh.ch`
♢ GNUnet e.V. `https://gnunet.org`

R^5N: Randomized-recursive routing for restricted-route networks

R^5N is a DHT with the following design goals:

- **Open participation peer-to-peer routing**.
- Works in **restricted-route environments**.
- Supports **route path recording**.
- In-band **request (and response) validation**.
- Allows for **result filtering**.

Open participation peer-to-peer routing

- Access control requires authentication (and trust) and leads to centralization.
- RELOAD (RFC 6940): "*RELOAD's security model is based on each node having one or more public key certificates. In general, these certificates will be assigned by a central server, which also assigns Node-IDs, although self-signed certificates can be used in closed networks.*"
- (Popular) DHTs today require classic Kademlia-style ad-hoc permissionless participation (e.g. IPFS).

Support for restricted-route environments

From "R^5N : Randomized Recursive Routing for Restricted-Route Networks" by Evans et al.:

- *Restricted-route topology* refers to a connected underlay topology which does not support direct connections between some of the nodes (e.g. wireless mesh networks, NAT or firewalls).
- Common DHT routing algorithms (e.g. Kademlia) show diminished performance or even arrant failure when operating over a restricted-route underlay.
- A common solution is to prevent participation in the DHT to peers that are not encumbered by such restrictions.
- However, on the modern Internet the proportion of hosts with unrestricted communication capabilities is increasingly limited (e.g. CG NAT).

Implications of restricted-route environments

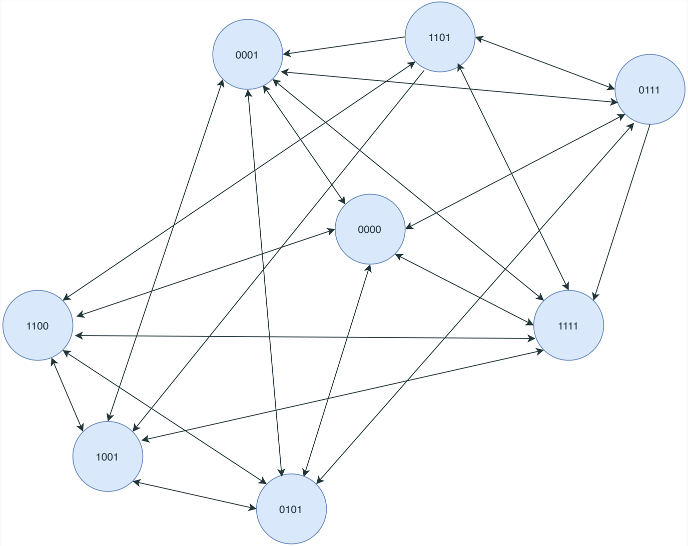
Problem:

- Some peers, which from the distance metric (XOR) may be close, may not be reachable (e.g. firewall).
- This leads to multiple (local) minima with respect to where data may be stored/can be retrieved.

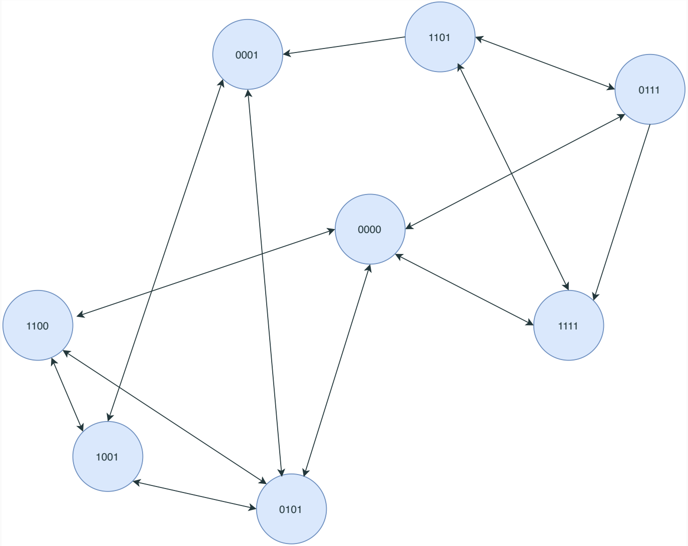
Solution:

- Random walk before greedy decent to "escape" local minima.
- Assuming we have a small world topology, the random walk will cause us to land at a random peer in the network from where the greedy descent will find a random local minimum.
- Replication at multiple local minima combined with the birthday paradox provides reasonable availability.

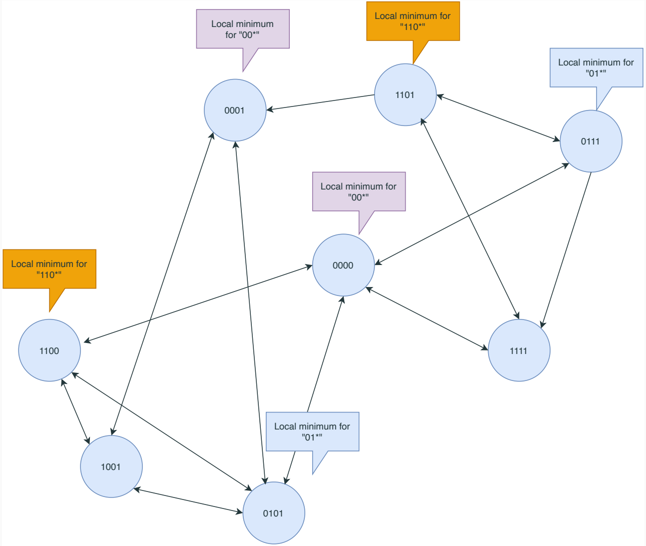
Kademlia sunshine scenario (k=2)



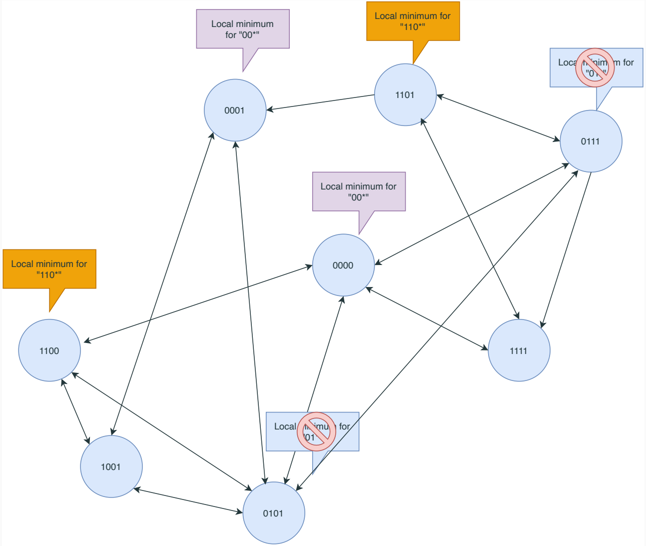
Restricted route scenario



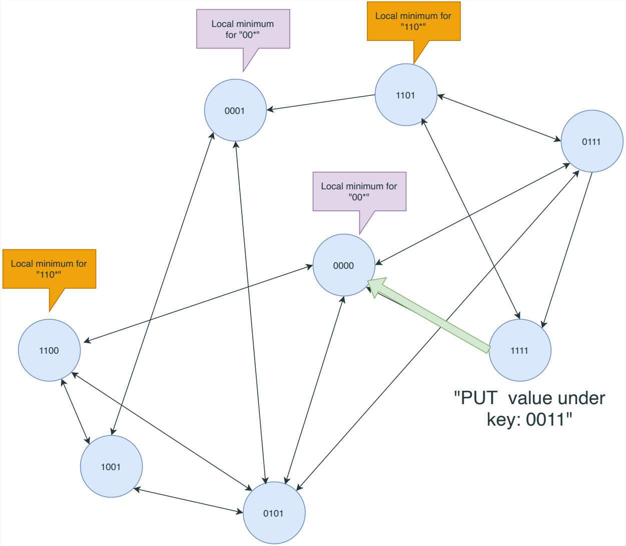
Local minima I



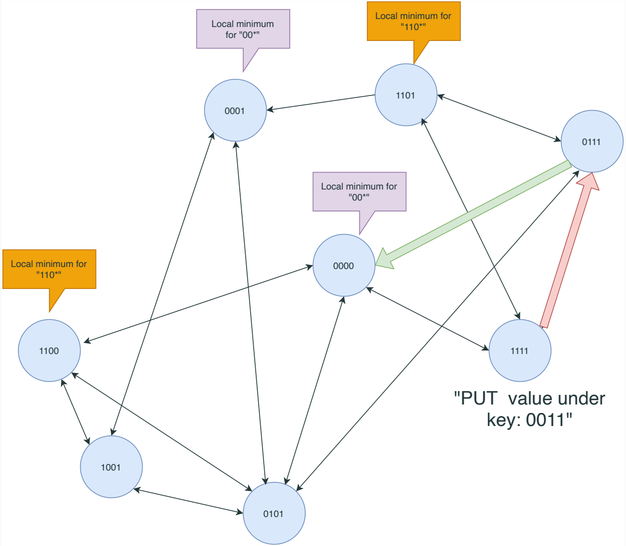
Local minima II



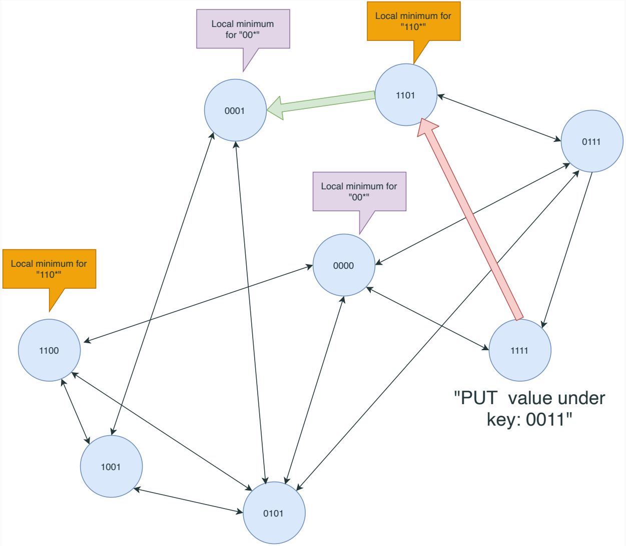
PUT example — XOR



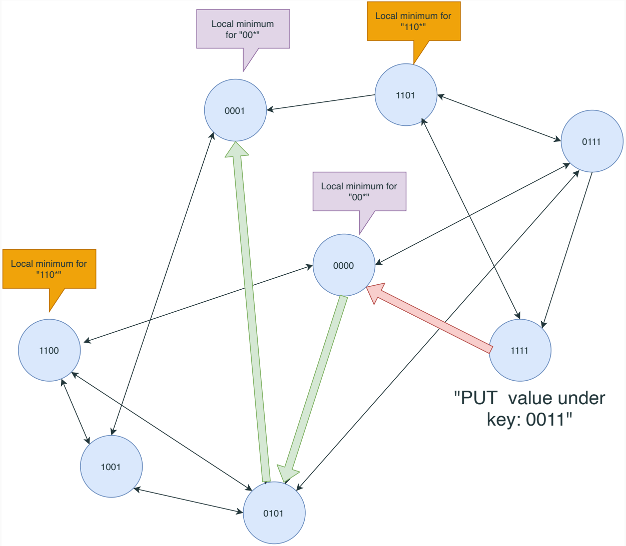
PUT example — R^5N walk length $= 1$



PUT example — R^5N walk length $= 1$



Special case: At least one descent-hop; no loops



Route recording for source routing

Consider the following problem:

- Two peers want to use a communication channel.
- They cannot establish a direct link due to underlay restrictions.
- Assumption: Other peers are happy to provide relay services.
- Payload transmission via PUT and GET would be inefficient.

⇒ Discover a route through the overlay:

- Peer adverstises existence of service via DHT PUT with route recording.
- Client discovers service provider via DHT GET with valid route of GET/PUT message path.

In-band response validation

DHT values can be corrupted or invalid. R^5N addresses this with pluggable, extensible block types:

- Given a key and a block type, it is possible to verify the integrity of the value.
- The verification should be possible for all hops on path, improving caching performance.
- A verification could include cryptographic signatures over the data or more sophisticated approaches (see GNS, RFC-to-be 9498)

Result filtering via mutated Bloom filter

Queries could have a unique or multiple results depending on the application.

- We provide capability to abort query forwarding early if unique answer has been found.
- We probalistically filter results already known to the client to reduce traffic.
- To address false positives when using Bloom filters we use mutation.

Optimization: Routing loop prevention

Repeatedly visiting the same peer in GET or PUT operations is inefficient.

- Visiting new peers increases the chance of finding previously undiscovered results.
- Visiting new peers drives us away from the starting point and towards more distant local minima.

R^5N uses a *Bloom filter* in GET/PUT messages to prevent routing loops.

For DISPATCH

- I-D is WIP at `https://datatracker.ietf.org/doc/draft-schanzen-r5n/`
- We have approached WGs since initial upload: dinrg, rtgwg, ...
- Which (other) WGs may be interested?

Funded by

NGI ASSURE

nlnet FOUNDATION

Contacts:

Martin Schanzenbach `schanzen@gnunet.org`

Christian Grothoff `grothoff@gnunet.org`