Pippolo

a NOSQL distributed DB

Features

- Pure XML communications
- Configurable high availability
- An always extensible graph
- Auto-sync between new Pippoli
- Completely written in C with POSIX libraries *(you can compile it everywhere)*

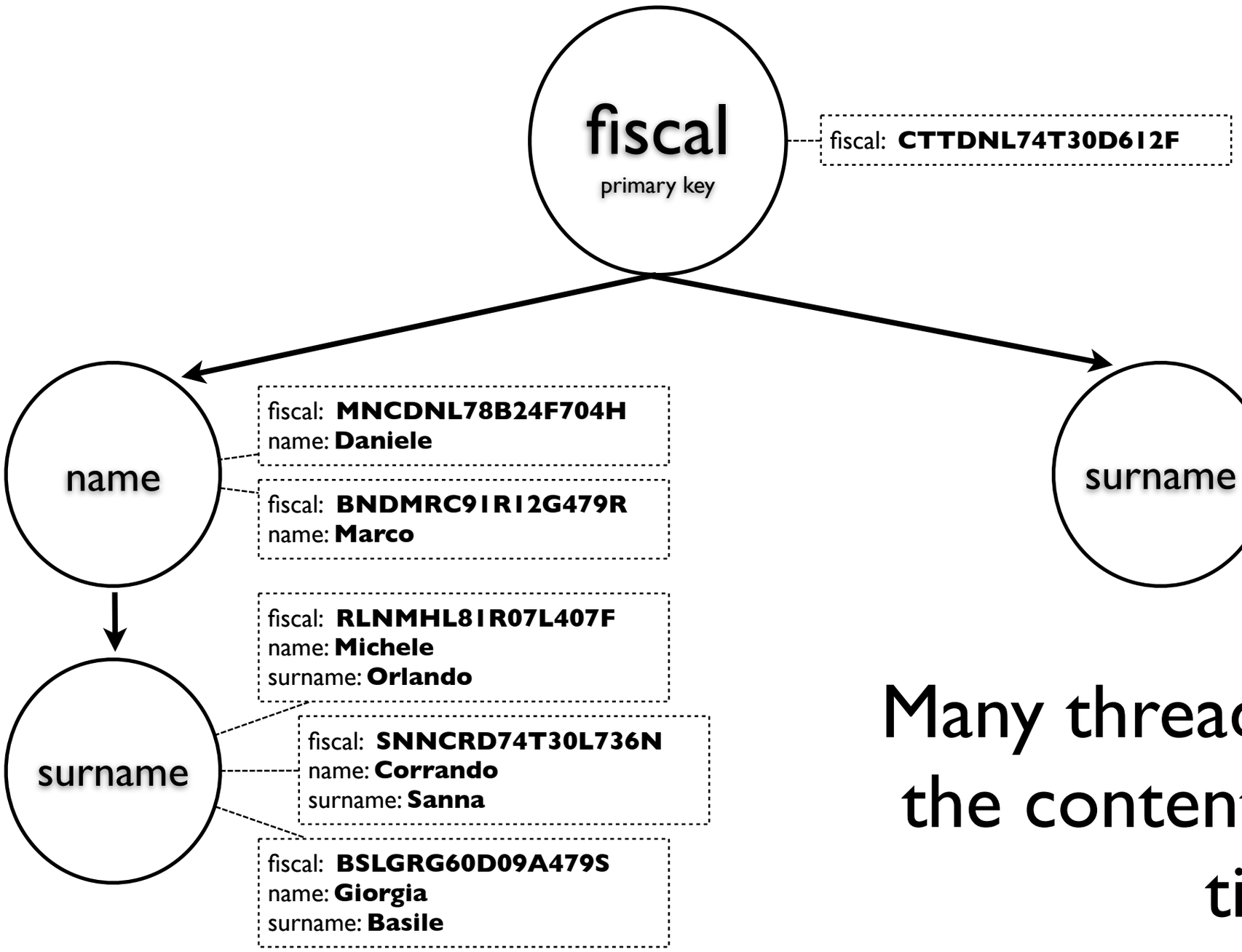
Features

- Each Pippolo has a covering range of 10 elements (from 0 to 9)
- Each record must be combine with an hash value (from 0 to 9) based on the primary key and calculated by the user using a personalized function
- Only those nodes whose hashes required by an action have been assigned to will execute the action

Features

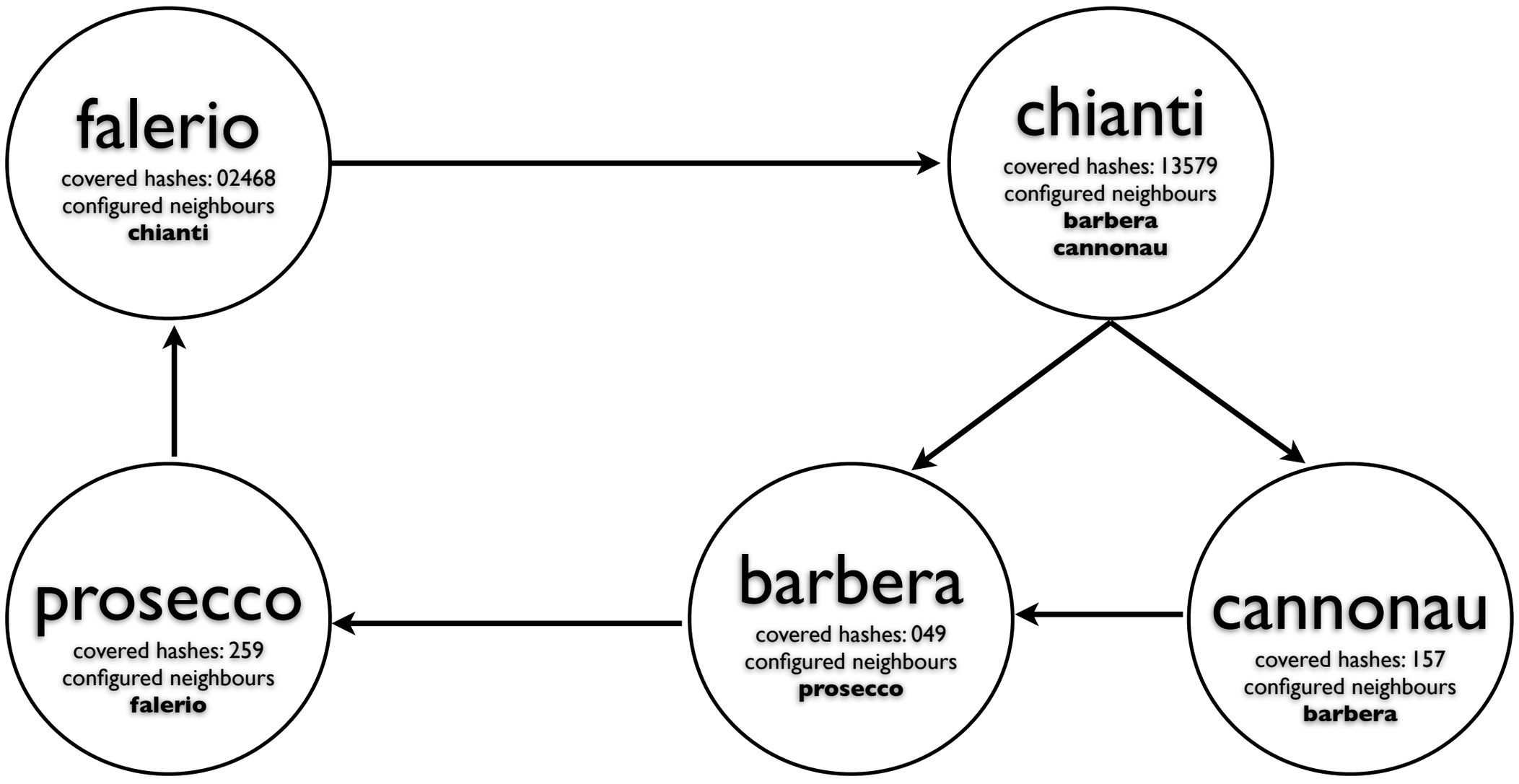
- Data is stored using trees *(you can perform parallel lookups by spawning threds to traverse different branches)*
- Regex based query
- Open source

Storing



Many threads can analyze the content at the same time
(not yet implemented)

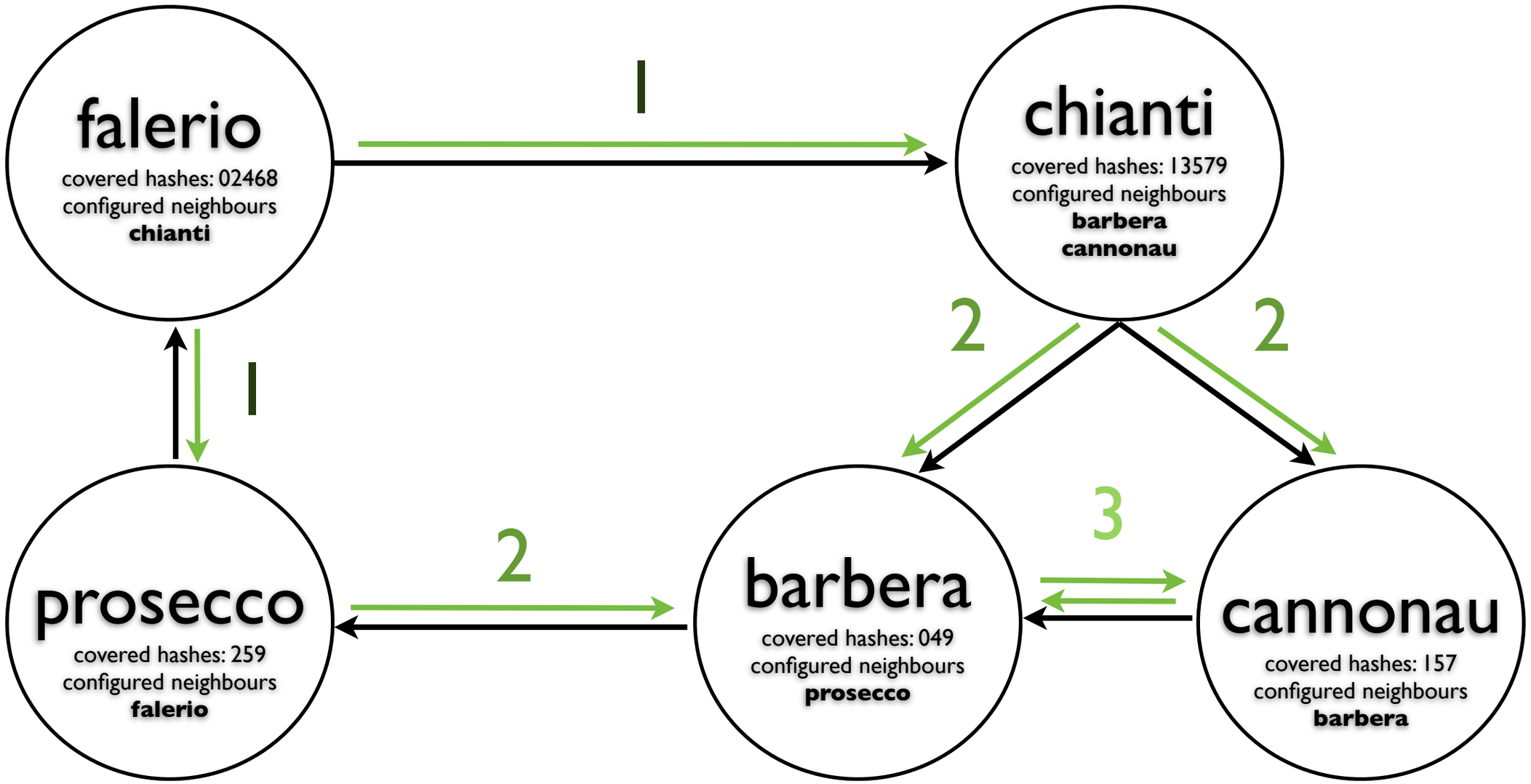
Example



Distributed Algorithm: **flooding**

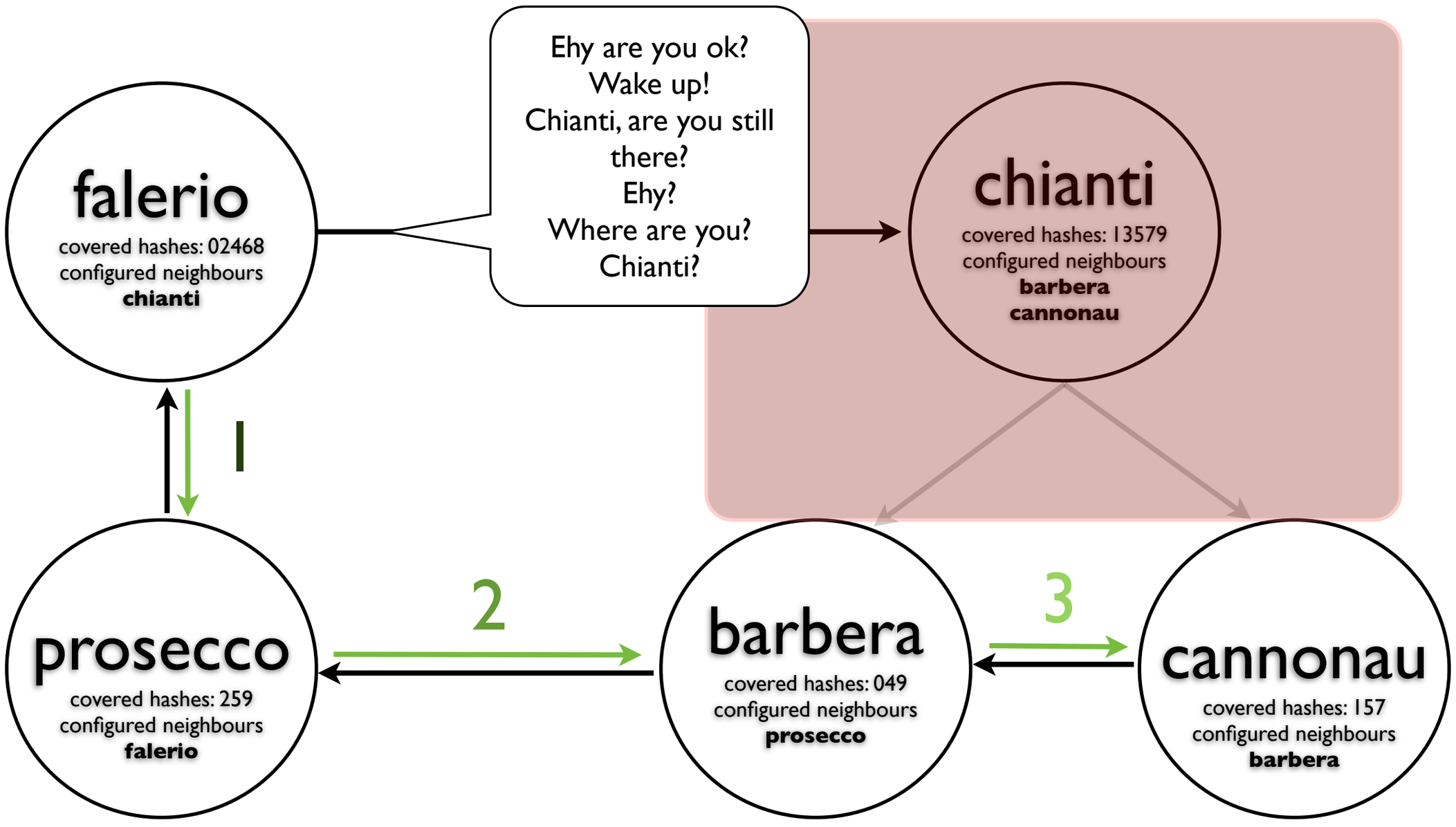
Example

add


falerio
covered hashes: 02468
configured neighbours
chianti
1
chianti
covered hashes: 13579
configured neighbours
barbera
cannonau
2
2
1
prosecco
covered hashes: 259
configured neighbours
falerio
2
barbera
covered hashes: 049
configured neighbours
prosecco
3
cannonau
covered hashes: 157
configured neighbours
barbera

Example

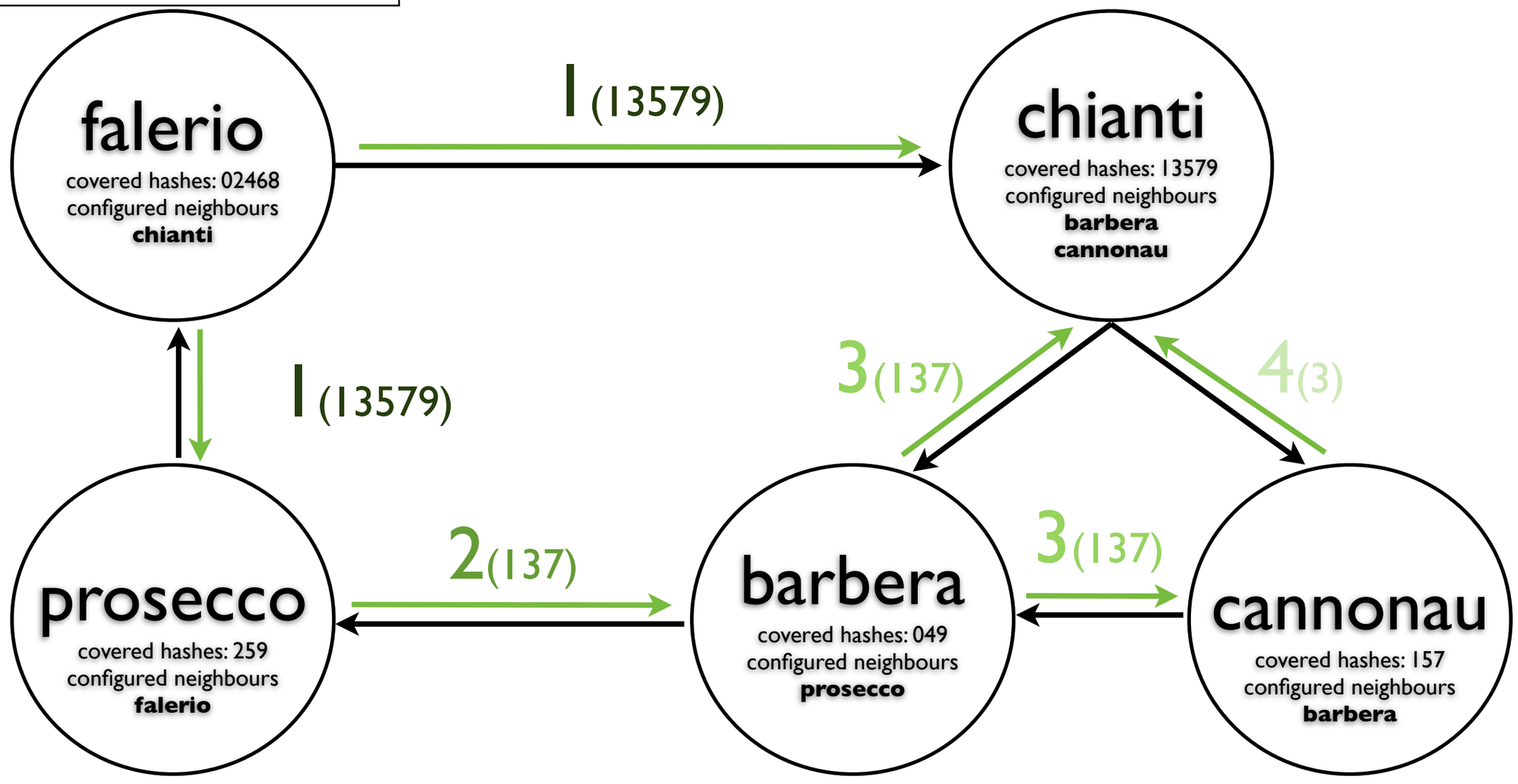
add


Ehy are you ok?
Wake up!
Chianti, are you still there?
Ehy?
Where are you?
Chianti?
falerio
covered hashes: 02468
configured neighbours
chianti
chianti
covered hashes: 13579
configured neighbours
barbera
cannonau
1
prosecco
covered hashes: 259
configured neighbours
falerio
2
barbera
covered hashes: 049
configured neighbours
prosecco
3
cannonau
covered hashes: 157
configured neighbours
barbera

Example

data request

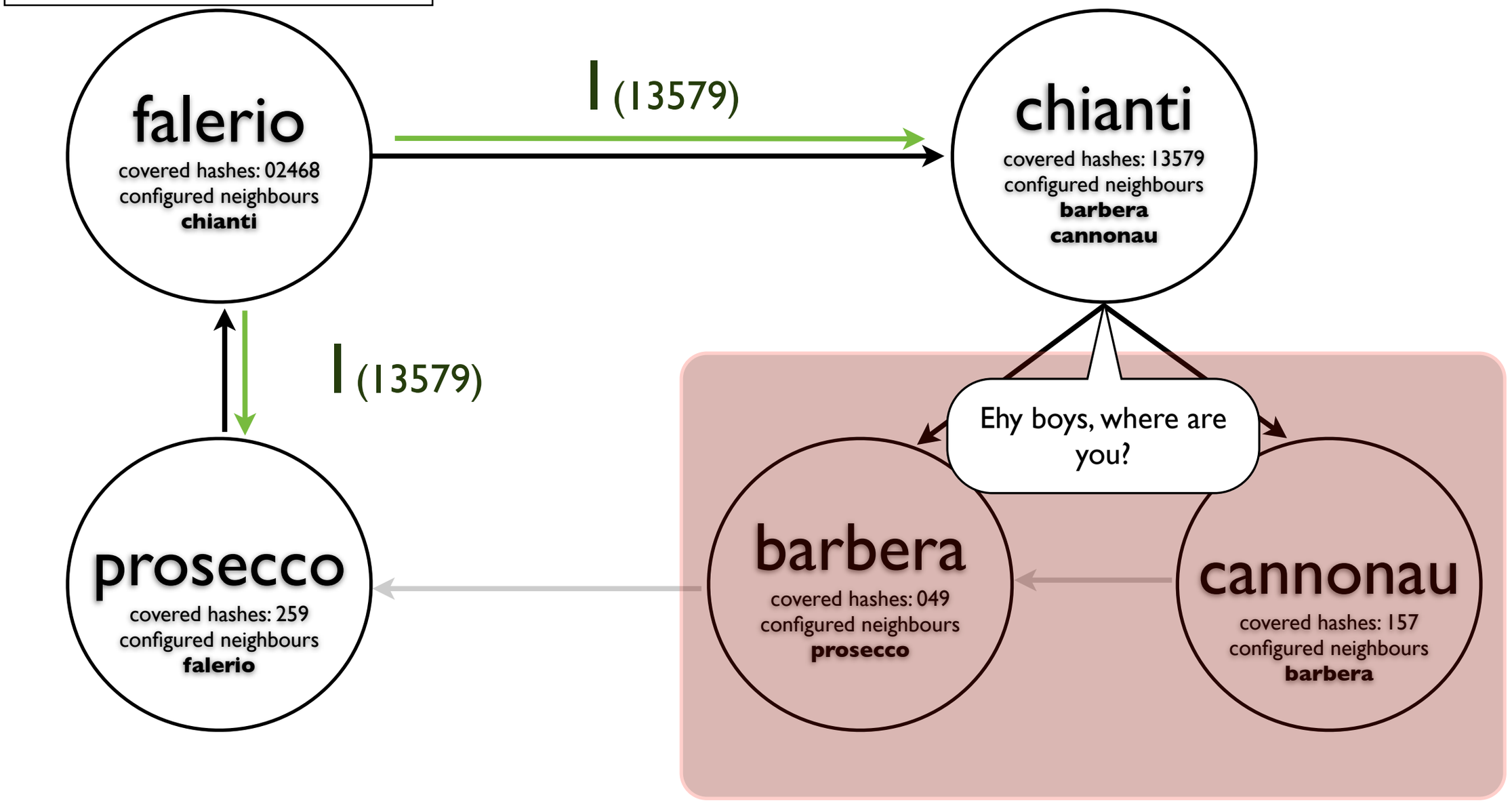
Incoming request for each record that cover hashes: 0123456789



Example

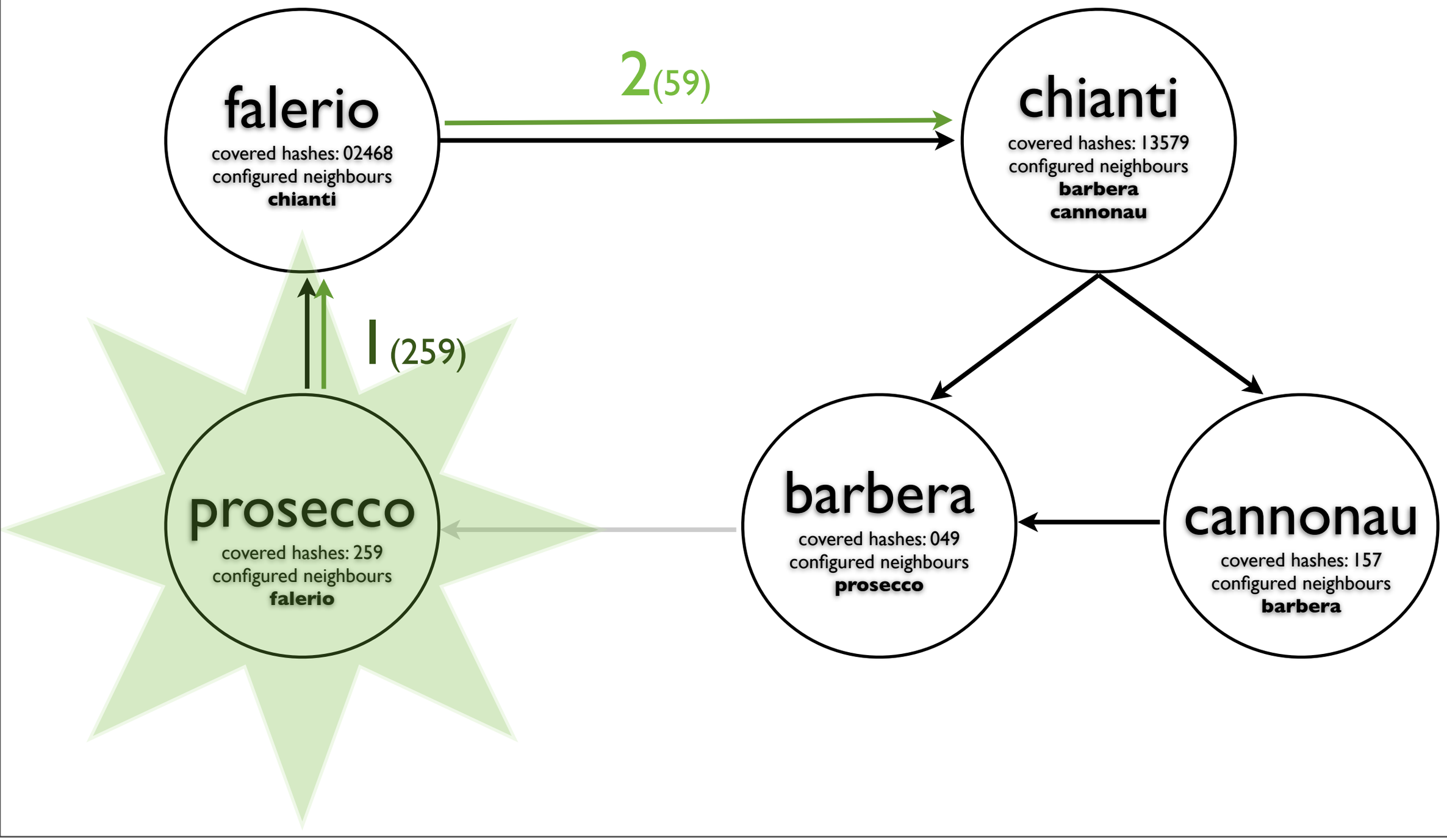
data request

Incoming request for each record that cover hashes: 0123456789



Example

wake up



Implementation

pippolapi

- Completely written in C with a personalized XML parser integrated
- Abstract
- Multiplatform (no dependences)
- Easy interfaced with any C/C++ projects

Example

add (by source)

```c
#include "pippolo_api.h"
void hooker (const char *ID, struct str_xml_node *result) {
    printf("operation %s complete\n", ID);
}

int discretizer (const char *value) {
    int result = 0;
    char *ptr = (char *)value;
    while (*ptr)
        result += *ptr++;
    result = (result%10);
    return result;
}

int main (int argc, char *argv[]) {
    if (argc <= 1)
        return 0;
    struct str_record *records = NULL;
    p_node_pippolo_init("vianello");
    p_node_pippolo_add("127.0.0.1", 5090);
    p_node_pippolo_add("127.0.0.1", 7090);
    p_node_pippolo_add("127.0.0.1", 4090);
    pippolo_discretizer = &discretizer;
    p_node_record_add(&records);
    p_node_record_keys_add(&records, "fiscal", "GRSDNL91R12A662K", pippolo_true);
    p_node_record_keys_add(&records, "name", "Daniele", pippolo_false);
    p_node_record_keys_add(&records, "surname", "Grassi", pippolo_false);
    p_node_action("inserimento unico", EDATA_ACTIONS_ADD, records, 10, &hooker);
    while (pippolo_true); /* waiting */
    return 0;
}
```

TODO list

TODO list

- Ability to dump the knowledge to files and data cacheing to support searching with previsioning algorithm
- Starting multithreading in the tree for data searching
- Replacing the flooding algorithm with a smartest algorithm (using service ports)

TODO list

- Authentication
- Safe communication protected with asymmetrical criptography (using an internal CA)
- CLI remote node control
- Update the configuration runtime