Logged in with account **rohan.prinja@gmail.com** - username: **rohanprinja** (logout)
Looking for Google Code-in? Click here!

MENU

- Home
- My Profile
- My Dashboard
- About
- Accepted Projects
- Events & Timeline
- Connect With Us
- Search
- FAQ
- Logout

Content of this proposal is not editable at this moment, because student application period is over. If you still need to make some changes, please send a comment to the organization. They can enable modifications for this proposal.

GNU Guix: Implementing a DHCP client in Guile Scheme

Rohan Prinja

View revisions

Organization: GNU Project

Abstract: We want a DHCP client written purely in Guile Scheme, that will be bundled as a Guix package, runnable as a GNU dmd service and integrated into dmd's proposed event loop.

Project name

Implementing a DHCP client in Guile Scheme

Summary

We want to have a DHCP client written in Guile Scheme. Specifically:
1. The implementation will comply with IETF RFC 2131, the RFC that describes DHCPv4 [1]
2. The client will be bundled as a Guix package [2], which installs a command-line program similar to ISC's dhclient (8) utility. [4,5]
3. The client code should be modular, so that some of its functions can be invoked from within GNU dmd's [6] proposed event loop.
4. The client should be runnable as a dmd service, as described in the dmd manual [6]
5. The code will be documented in texi format, which can then be converted to info pages (and other formats too).

Bonus

As a bonus task for the project, the module should support DHCPv6 (described in IETF RFC 3315 [9]). That is, the code should implement both versions of the DHCP protocol.

The tasks in the 'Summary' section above will be completed two weeks before the final date, including documentation and deployment to Guix and integration with dmd. The remaining two weeks are reserved for extending the module to support DHCPv6. I will try to keep the code general enough while writing the DHCPv4 implementation, so that there is less overhead in the bonus part.

Communication

Name: Rohan Prinja
Email: rohan.prinja@gmail.com
IRC: wenderen (#guix)
Github: wenderen
Mobile: +91 96190 19345
Country of residence: India
Timezone: IST (GMT +530)
Primary Language: English

I will be present on IRC (#guix), throughout the course of the project for communication, weekends included. In addition, I can always be reached via e-mail or phone (if needed).

Benefits

Currently the only DHCP client implementation in the Guix package list is the ISC implementation, which is written in C. The aim of this project is to deliver a Guile Scheme implementation of a DHCP client satisfying the conditions outlined in the abstract. The motivation for this is that GNU dmd should be able to easily call DHCP library functions from its event handlers in response to various network events. Since dmd is written in Guile Scheme, we want our DHCP library to be written in Guile Scheme as well.

How users benefit

1. This project is related to dmd gaining an event loop so that it can properly handle events - including DHCP events - rather than simply waiting for the client. dmd performing faster and better is a win for the user.
2. More generally, since one of the deliverables is a binary, users of Guix SD and in general any architecture which is a compile target for Guile can use the DHCP client.

How GNU benefits

Guix SD gains an important daemon.

Deliverables

1. A Guile Scheme module, implemeting RFC 2131, and for bonus, RFC 3315.
2. A command-line program installable via the Guix package manager, which calls the module's functions.
3. A dmd service for the DHCP client.
4. A demonstration of how to call the client module functions from within an event loop in dmd in response to network events.

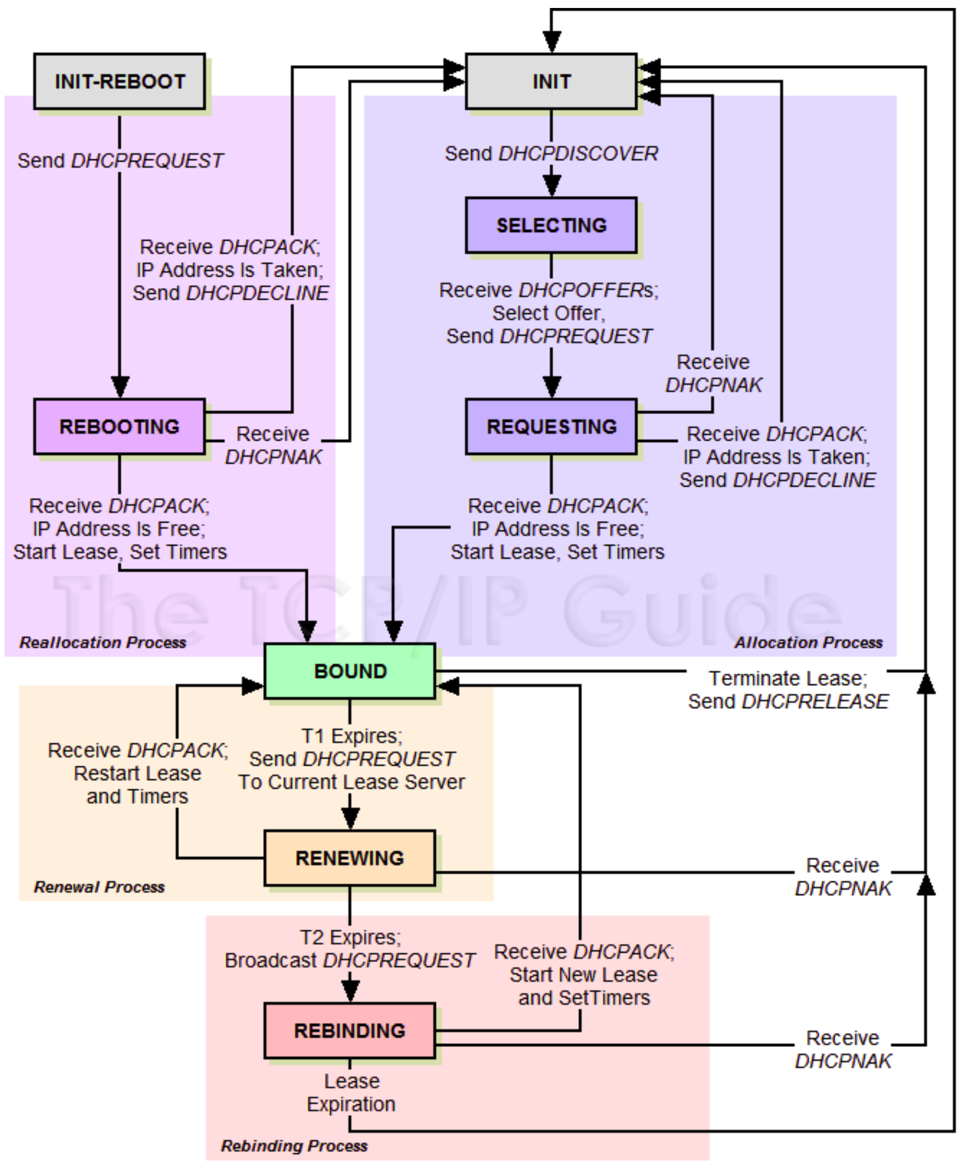
Notes

1. As of the time of proposing this project, dmd does not have an event loop. So, deliverable 4 will involve:

- Adding a simple event loop to dmd for testing purposes (a full-fledged event loop is out of the scope of this project)
- Setting the handlers for network events like lease expiration to be functions from the client module

2. Implementing deliverable 1 is a prerequisite for implementing 2, 3 or 4. Each one of 2, 3 and 4 act as real-world test cases for 1.

DHCP protocol


INIT-REBOOT
Send DHCPREQUEST
Receive DHCPACK; IP Address Is Taken; Send DHCPDECLINE
REBOOTING
Receive DHCPNAK
Receive DHCPACK; IP Address Is Free; Start Lease, Set Timers
Reallocation Process
INIT
Send DHCPDISCOVER
SELECTING
Receive DHCPOFFERs; Select Offer, Send DHCPREQUEST
Receive DHCPNAK
REQUESTING
Receive DHCPACK; IP Address Is Taken; Send DHCPDECLINE
Receive DHCPACK; IP Address Is Free; Start Lease, Set Timers
Allocation Process
The TCP/IP Guide
BOUND
Terminate Lease; Send DHCPRELEASE
Receive DHCPACK; Restart Lease and Timers
T1 Expires; Send DHCPREQUEST To Current Lease Server
RENEWING
Receive DHCPNAK
Renewal Process
T2 Expires; Broadcast DHCPREQUEST
Receive DHCPACK; Start New Lease and SetTimers
REBINDING
Receive DHCPNAK
Lease Expiration
Rebinding Process

The DHCP protocol describes a series of steps to be taken by a host (client) connected to a network when it requires an IP address. The idea is that a set of servers, called DHCP servers each maintain a disjoint pool of IP addresses which they lease out to clients who are connected to the network. The high-level view of the protocol is that the client broadcasts a message requesting an IP address, some of the DHCP servers respond by offering IP addresses, and the client selects one and responds with a request for that specific address. The targeted server then confirms this address and the client is allocated the IP address. The server and client do some bookkeeping to store the IP address allocation. The protocol is further complicated by the fact that messages may be lost, or that servers may allocate incorrect addresses. The RFC that introduced the DHCP protocol anticipated these issues and dealt with them in various ways, such as timers which indicate when a lease is about to expire.

High-level plan

This project involves writing a library more or less from scratch. So designing and discussing the structure of the library with knowledgeable community members is a prerequisite. I've tried to keep this task before the actual GSoC period, because I want to reserve the GSoC period for writing code, code review and committing.

I propose the following high-level timeline:

Pre-GSoC Period (before May 25)

- Environment setup, finish all background reading
- Finalize design of client library API

- Finalize design of CLI program

GSoC Period

Phase 1 - Main deliverables (May 25 - Aug 9)

- Write a stubbed DHCP client module which exports skeleton functions (for example, a skeleton function might simply print to stdout). This will include:
  - functions for constructing and sending DHCP packets over a network.
  - functions that handle client logic, like transitioning between states in the FSM [8].
  - functions for writing to and reading from and the local store.
- Write the command-line program and call the module's functions from within it.
- Call the module's functions from within the event loop and ensure that the functions are being invoked correctly in response to events like DHCP lease expiration, for example.
- Fill the stubs. This involves making use of Guile Scheme's networking library [7] and referring to the dhclient source [4].
- Test the completed command-line program
- Create and test a dmd service for the client.
- Patch dmd with a bare-bones event loop so that we can test events related to the DHCP client process.
- Test the integration with dmd's event loop after the stubs are filled in.

Phase 2 - Bonus (Aug 2 - Aug 16)

- Add functions to the module to support DHCPv6 as well as DHCPv4.

Post-GSoC Period (after Aug 17)

- Document the DHCPv4 client module
- If bonus was successfully completed:
  - Document the DHCPv6 client module
  - Else, continue the implementation of DHCPv6 client module and document once finished
- Continue involvement with community
  - Further patches, bug fixes, maintaining DHCP client library

Documentation

In accordance with the GSoC guidelines, I am relegating documentation and cleaning up the code to the 'pencils down' period, while the main period (May 25 - August 17) is reserved for writing code, committing and sending patches.

Detailed plan + Midterm evaluation

The above section was a high-level plan. In this section the detailed weekly plan is given. Although not mentioned explicitly below, I am setting aside 2 hours each weekend to blog about my weekly activity. This will work as a record of my progress as well as a reference for interested people. Also, the end of each working week includes code review, in which I will discuss submitted patches with the community members and make changes as needed.

April 27 - May 25 (Pre GSoC period)

- Read DHCPv4 [1] and DHCPv6 [9] RFCs, take notes (already started)
- Learn and play around with Guile Scheme's module system and networking library
- Set up and play with dmd on my machine
  - Starting and stopping services, querying running services
  - Learn how to add a service to GuixSD and dmd
- Design API of client module, discuss with mentor(s) and receive feedback
- Discuss possible structure of dmd event loop with mentor(s), prepare rough design, seek approval and finalise design
- Study ISC's dhclient source code [4] and design CLI program's API
- Finalize module functions and command-line program API

May 25 - May 31

- Write skeleton of standalone module, exposing functions like:
  - (construct-dhcp-message)
  - (send-dhcp-message)
  - (make-fsm-transition)
  - (store-local-lease)
  - (delete-local-lease)

    - (start-t1-timer)
    - (start-t2-timer)
- Commit skeleton module, send a patch to the mailing-list for review
- In the following weeks, these function stubs will be filled in. The idea is to have a top-down view of how the module will look as soon as possible.

June 1 - June 7

- Implement constructing and filling information in DHCP packets as per message type
- Implement parsing of received DHCP packets
- Implement sending of ARP packets (used by client to prevent address conflicts)
- Test sending and receiving of packets from client side
    - Sending: DHCPDISCOVER, DHCPREQUEST and DHCPNAK
    - Receiving: DHCPOFFER and DHCPACK
- Commit DHCP message sending/receiving and constructing/parsing code, send a patch to mailing-list for review

June 8 - June 14

- Begin writing skeleton command-line program (by referring to the design made in the pre-GSoC period)
    - Handle parsing command-line args and dispatching appropriate functions
    - Call out to module functions stubs from command-line program
- Commit skeleton command-line program, send a patch to mailing-list for review

June 15 - June 21

- Implement functions to interface with persistent storage:
    - Create storage file if it doesn't exist with appropriate permissions
    - Write lease to persistent storage
    - Read from storage and set host's IP address and network parameters
- Implement lease timers T1 and T2
- Commit storage functions and timer code, send a patch to mailing-list for review

June 22 - June 28

- Implement client-side finite state machine. This consists of the following sub-tasks:
    - Create states as per the references [1], [8]
    - Add state transition logic. In DHCPv4, state changes are either due to:
        - receipt of a DHCP message
        - or expiry of a lease timer
    - Call the previously-written functions for sending/receiving messages, interfacing with storage and timer logic from within the appropriate states
- Test that the transitions happen as expected
- Commit state machine code, send a patch to mailing-list for review

June 29 - July 3

- Now that the DHCP client functions have been written, the command-line program is also complete
- Stress-test command-line program
- Bundle as a Guix package and upload to the package store
- Test that the package is downloadable and installable

July 3 (Midterm Evaluation)

Fully functional command-line program. Analogous to ISC's dhclient, the user should be able to use it to:

- Obtain IPv4 address + configuration parameters
- Obtain and release a lease

In addition, it should support the various switches and flags that dhclient does, including -p (specifying port number), -s (specifying server address), -e (setting environment variables). However, unlike dhclient, it cannot (as of the midterm evaluation) support the -6 flag (IPv6 addresses).

July 6 - July 12

- Write a dmd service that calls out to the command-line program
- Add it to GuixSD's list of services [12]

- Test that the service works with dmd and deco
  - The service should be stoppable and startable
  - deco should be able to query its status and detailed-status as explained in the dmd manual [6]
  - deco should be able to perform actions specific to the service, in this case: calling the command-line program with any of its flags
- Commit changes, send a patch to mailing-list for review

July 13 - July 26

- Replace straightforward event-loop [15] in dmd with more robust version based on discussions with mentor(s) and community members
- Test dmd to ensure that no regressions occur after the loop replacement
- Commit the new loop implementation, send a patch
- Identify which network/other events are the responsibility of the DHCP client. For example:
  - Timer event due to lease expiry
  - Receipt of a DHCP message from a DHCP server on the subnet
- Register the functions of the client module as handlers for these events
- Test that the event loop correctly invokes the handlers
- Commit the added handlers, send a patch for review

July 27 - Aug 2 (Buffer period)

- Buffer period: Reserved for running further tests on all of the code written thus far, refactoring if needed, responding to bugs encountered when using the client
- Commits/patches: as frequently as required (depends on extent of refactoring/testing)

Aug 3 - Aug 16 (Bonus)

- Extend client module functions to support DHCPv6
- Add -6 flag to command-line program
- Similarly, modify dmd service and event loop handlers to support v6 changes
- Re-test command-line program, dmd service and dmd event loop integration
- Commit v6 additions, send patch to mailing-list for review

August 17 (Soft 'pencils down')

- Write .texi documentation for client module, command-line program and dmd service
- Publish .texi docs to info pages and man (to convert to man, I propose using GNU wget's method: run a Perl script [13] to convert .texi to .pod and then running pod2man on the .pod file)
- Commit the documentation, send patch to mailing list

August 20 (Firm 'pencils down') onwards (Post-GSoC period)

- Continue writing documentation and further tests, if not finished by 'soft pencils down' date
- Maintain DHCP client, respond to bugs encountered by users, if any
- Contribute to other parts of the GuixSD codebase
- Help newcomers on IRC/mailing lists work with Guix and dmd

Commitments

I graduate in early May and join my place of employment in early September. I have no commitments in between these two dates, except for taking two days off in early August to attend my graduation ceremony (the precise date has not been announced by my college authorities).

So, I can work for at least 40 hours every week, and put in more time if needed to meet the above deadlines. I am very serious about delivering a good piece of software that works as it should and is a useful addition to the Guix package archive!

Qualification

Why does this project appeal to me?

I am interested in this project because:

1. I want to do impactful work, and a DHCP client is a vital component of modern operating systems.
2. Writing a standards-compliant library looks like an interesting challenge.

3. I did a "Networks" course in college and enjoyed it. I also have contributed a few small patches to Hyper [10], the HTTP library used by the Servo browser, as well to Servo itself [11]
4. I'm interested in programming in Guile Scheme. I did an "Abstractions and Paradigms in Programming" course in college, where I programmed in Racket. It was fun and when I encountered this project I jumped at the chance to work with a Lisp-family language.

Relevant experience

1. I have contributed small patches to Google Skia as part of an internship with Samsung, and, as already mentioned above, have some commits to Mozilla Servo [16] and Hyper (on a voluntary basis).
2. As mentioned above, I have experience in network programming and am familiar with Racket. In my network programming course, I wrote a toy distributed file server.
3. I am comfortable with using mailing lists, responding to code reviews and giving and asking for help on IRC.
4. I have made a small patch to GNU wget [14], so I am familiar with the patch/mailing-list workflow model that the Guix and dmd community use.

What will I do after GSoC ends?

I see GSoC as a stepping stone to being actively involved in the community by contributing and maintaining code, and also bringing in newcomers to GNU projects, in particular Guix. If I am selected and the project is very successful, I would like to apply as a mentor in the following years.

Skills needed

Apart from the network knowledge mentioned above (for example DHCP protocols), I will need to:

- Know how to idiomatically use the Guile Scheme ecosystem (writing and designing modules)
- Become familiar with the Guile Scheme POSIX networking API

Why am I applying to GNU?

I’m grateful to GNU both for creating software that I use everyday, and for promoting a philosophy of software that respects the user. I’m very excited to be applying to GNU. I see this as a chance to do two things:

1. Do a useful and impactful project for a community that has given me so much.
2. As mentioned above, I want to join the community in earnest, using this project as a first step.

References

[1] RFC 2131, which describes DHCPv4: https://www.ietf.org/rfc/rfc2131.txt
[2] Guix package manager: http://www.gnu.org/s/guix
[3] List of available Guix packages: http://www.gnu.org/software/guix/package-list.html
[4] ISC DHCP implementation: https://www.isc.org/downloads/dhcp/
[5] dhclient man page: http://linux.die.net/man/8/dhclient
[6] dmd manual: http://www.gnu.org/software/dmd/manual/dmd.html
[7] Guile networking: https://www.gnu.org/software/guile/manual/html_node/Networking.html
[8] DHCP FSM: http://www.tcpipguide.com/free/t_DHCPGeneralOperationandClientFiniteStateMachine.htm
[9] RFC 3315, which describes DHCPv6: https://www.ietf.org/rfc/rfc3315.txt
[10] Hyper HTTP library: https://github.com/hyperium/hyper
[11] Servo is a browser under development by Mozilla and Samsung: https://github.com/servo/servo
[12] GuixSD services are defined in the networking.scm source file: git.savannah.gnu.org/cgit/guix.git/tree/gnu/services/networking.scm
[13] texi2pod.pl, a script to convert .texi to .pod which can then be further converted to man pages: http://git.savannah.gnu.org/cgit/wget.git/tree/doc/texi2pod.pl
[14] The patch added UTF-8 support to the documentation for wget. This solves the problem of many contributors' names showing up incorrectly in man pages due to the presence of non-ASCII Latin alphabet characters such as č or ä. Link to patch: http://git.savannah.gnu.org/cgit/wget.git/commit/?id=8d4bb928b921b7620f1b3d61f04ed848138c3d7c
[15] As of now, dmd's event loop simply waits for the a command from clients, which could possibly stall dmd indefinitely: http://git.savannah.gnu.org/cgit/dmd.git/tree/modules/dmd.scm#n202
[16] Commits to Servo: https://github.com/servo/servo/commits?author=wenderen, Commits to Hyper: https://github.com/hyperium/hyper/commits?author=wenderen

2 comments

Ludovic C. March 27, 2015, 9:17 p.m.

Thanks for the extended proposal! The detailed plan makes sense, and the overall project looks good to me.

Rohan Prinja March 28, 2015, 5:10 a.m.

Thank you!

Leave a comment

Submit

- About
- Contact
- Google+
- Blogger
- Email
- IRC
- Privacy Policy

- Powered by melange
- Empowered by echoditto

Version 2-1-20150429 (Report Bug)