Curl user poll 2015

“retina-burning blue”

Daniel Stenberg, May 26, 2015

Table of Contents

Curl user poll 2015 – analysis

All analysis and conclusions drawn in this document are made by Daniel Stenberg. I do not speak for anyone else than myself.

About curl

Curl is an open source project and produces the curl tool and the libcurl library. We are a small project, with few core contributors with little commercial backing. Our products still run in a vast amount of internet-connected devices and services on the current Internet.

See curl.haxx.se for everything not answered in this summary.

Background

With this year's poll, we have established an annual tradition that hopefully will continue in the coming years. One of the primary ideas behind doing it annually is to take advantage of watching how the responders vary from year to year and use that as a crude method of detecting trends among our users. To allow that, we of course need to keep the questions as similar as possible to make them comparable.

In 2014 we did the poll in June, but I wanted to move it slightly earlier in the year mostly for personal reasons. The poll was available online for 10 days this year (from May 6th to May 16th), the exact same length as last year. The idea being that if someone is away for a week, it will still allow such a person to participate – and then some.

On the questions

I posted the planned questions weeks before the poll actually started with extremely little feedback given. Once it went live and in poll submissions we received feedback and comments on particulars in the survey. Things we could do differently. This is all natural and I'm convinced that no matter what, we would always get comments, praise and complaints. All in all, I'm happy with both the questions and the amount of response we have got.

Details to consider for next year's poll: make it more clear that to can skip questions and provide more “I don't know” alternatives.

Responses

In 2014 we received 193 responses.

In 2015 we received 1475 responses.

The huge difference in participation can probably only be attributed to this year's poll being distributed more widely across social media, and then in particular Twitter.

A question when doing this kind of survey of course always remain: do we actually get answers about the project or do the answers mostly show what kind of audience we managed to get answers

from?

Actions based on this poll

Deciding how to interpret the answers and making an action plan based on the results and conclusions is not easy. Especially since we all work on what we want, we all volunteer here and we can't force anyone to work on anything specific. My hope is still that all the details and my occasional “conclusions” in this analysis report will at least help us keep these facts in the back of our minds when we go forward.

Please check the protocols you use curl/libcurl for

This question allowed for all 21 supported protocol versions to be individually checked.

DICT	10	0.7%
FILE	79	5.4%
FTP	443	30.2%
FTPS	138	9.4%
GOPHER	22	1.5%
HTTP	1439	98.1%
HTTPS	1384	94.3%
IMAP	24	1.6%
IMAPS	26	1.8%
LDAP	26	1.8%
LDAPS	22	1.5%
POP3	20	1.4%
POP3S	15	1%
RTMP	19	1.3%
RTSP	13	0.9%
SCP	84	5.7%
SFTP	126	8.6%
SMTP	33	2.2%
SMTPS	24	1.6%
TELNET	37	2.5%
TFTP	44	3%

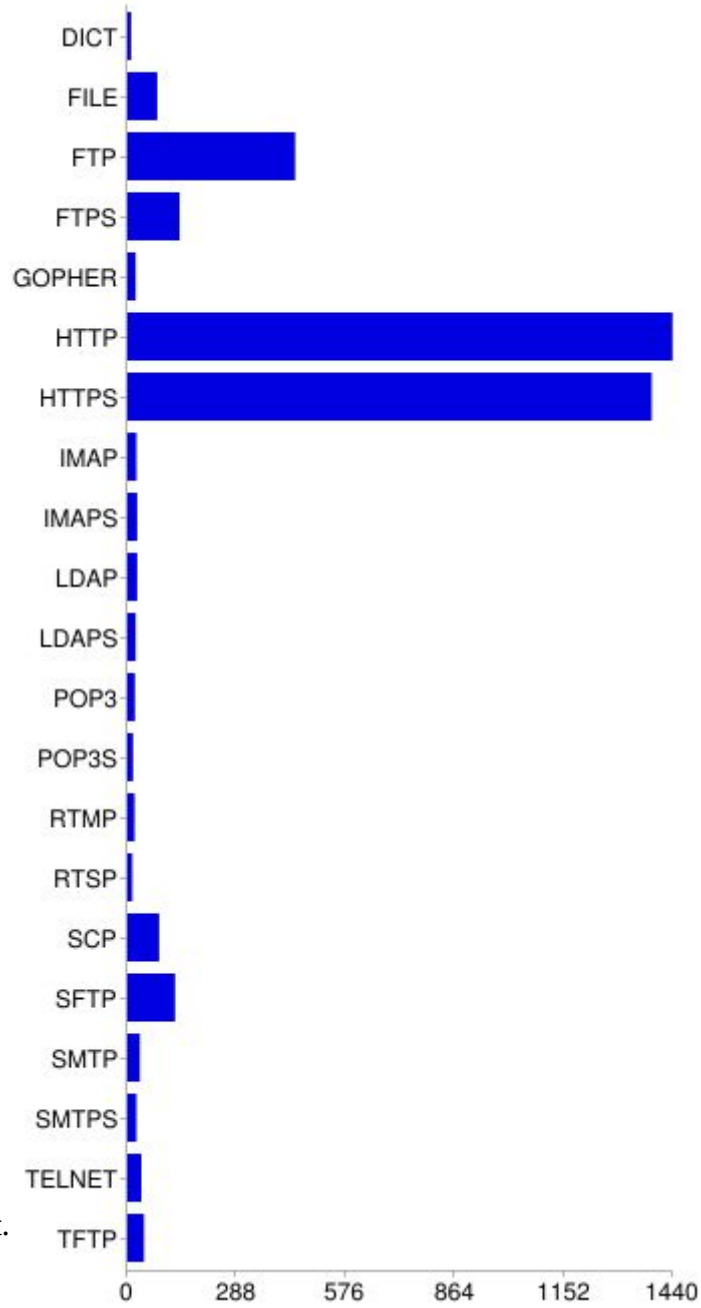


Illustration 1: Protocol popularity distribution

This set of results is not terribly different compared to last year, but what we can see is that HTTP and HTTPS are gaining even more ground this year and the relative popularity of the other protocols are all shrinking somewhat.

The popularity order among the top-5 remains the same too:

1. HTTP
2. HTTPS
3. FTP
4. FTPS
5. SFTP

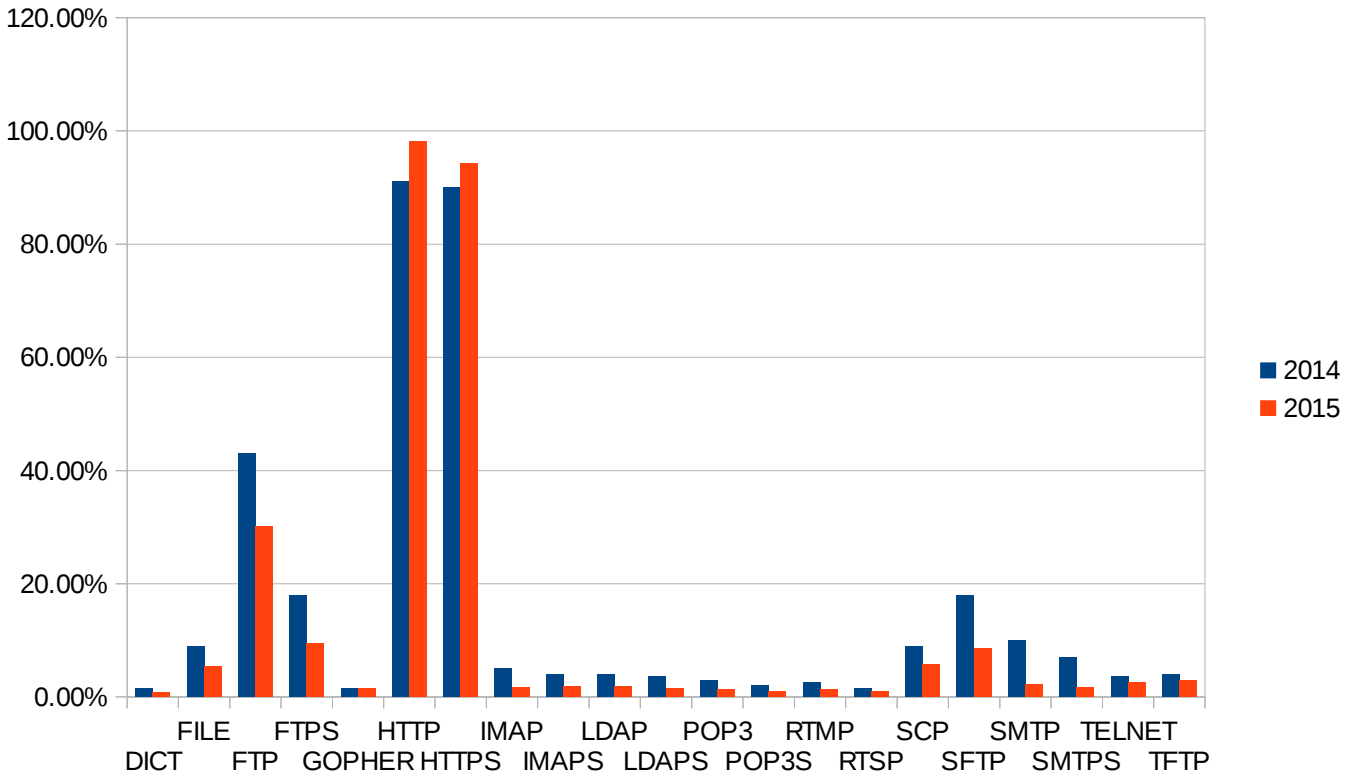
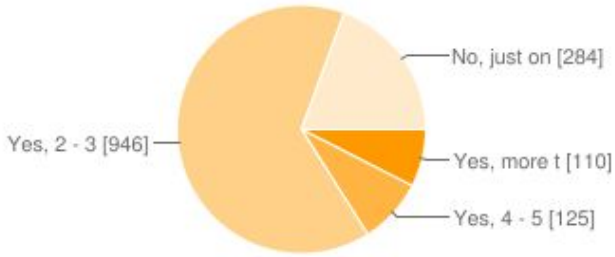


Illustration 2: The 2014 protocol usage distribution compared to the 2015

Do you use curl/libcurl on multiple platforms?

Remains basically the same as in 2014. 65% says 2-3 platforms, 19.4% just one, 8.5% 4-5 platforms and 7.5% claims working with it more than 5 platforms.

Last year, the numbers were 58/19/12/11 in the same order so basically the the number of 2-3 users are now larger, the single system share is the same and there are fewer user with 4 or more platforms.

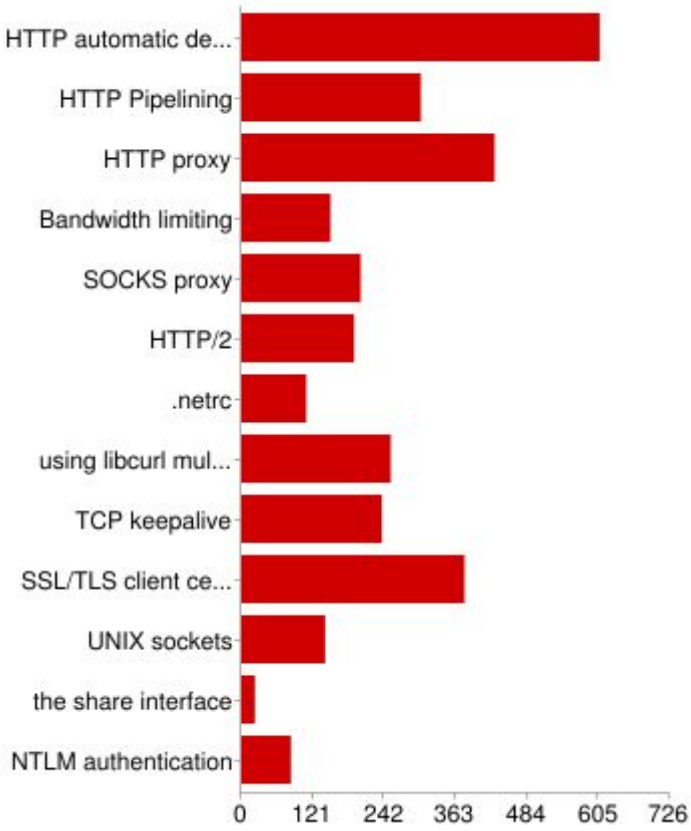


Tell us which libcurl features you use?

This is a new question for the year and of course the idea here is to sample the interest and usage share among a number of various features.

Recall that 1475 submissions were made in total:

HTTP automatic decompression	607
HTTP Pipelining	304
HTTP proxy	429
Bandwidth limiting	151
SOCKS proxy	202
HTTP/2	191
.netrc	110
using libcurl multi-threaded	253
TCP keepalive	238
SSL/TLS client certificates	378
UNIX sockets	142
the share interface	23
NTLM authentication	84



In general these results surprise me by the shear numbers. 45% of all users use automatic decompression!

20% use HTTP pipelining? The lack of bug reports for this still fairly complicated feature actually suggests that people check this more often than what they actually use. Pipelining with curl also only works for users of the libcurl API (either plain C or with a binding).

The mere 1.6% users of the share interface shows that we don't need to be too concerned about that being a bit left out recently.

A whopping 13% of the users already say they use HTTP/2 with curl.

Which SSL backend(s) do you typically use the most?

We do make an effort to support a wide variety of TLS/SSL library backends. Is this backed up by users actually appreciating this fact and thus using many different ones? Nope. This is the answer distribution. This table shows the percentage among the ones who actually answered this question, thus slightly higher than if we count as a share among the total number of responders.

OpenSSL	1286	97.1%
NSS	59	4.5%
GnuTLS	219	16.5%
libressl	79	6%
BoringSSL	4	0.3%
PolarSSL / mbedTLS	17	1.3%
CyaSSL / WolfSSL	6	0.5%
axTLS	1	0.1%
Securetransport / DarwinSSL (Max OS/iOS)	85	6.4%
schannel / WinSSL (windows)	61	4.6%
gskit (OS/400)	3	0.2%

I would've guessed that OpenSSL would be the biggest, but I didn't foresee this landslide “win”.

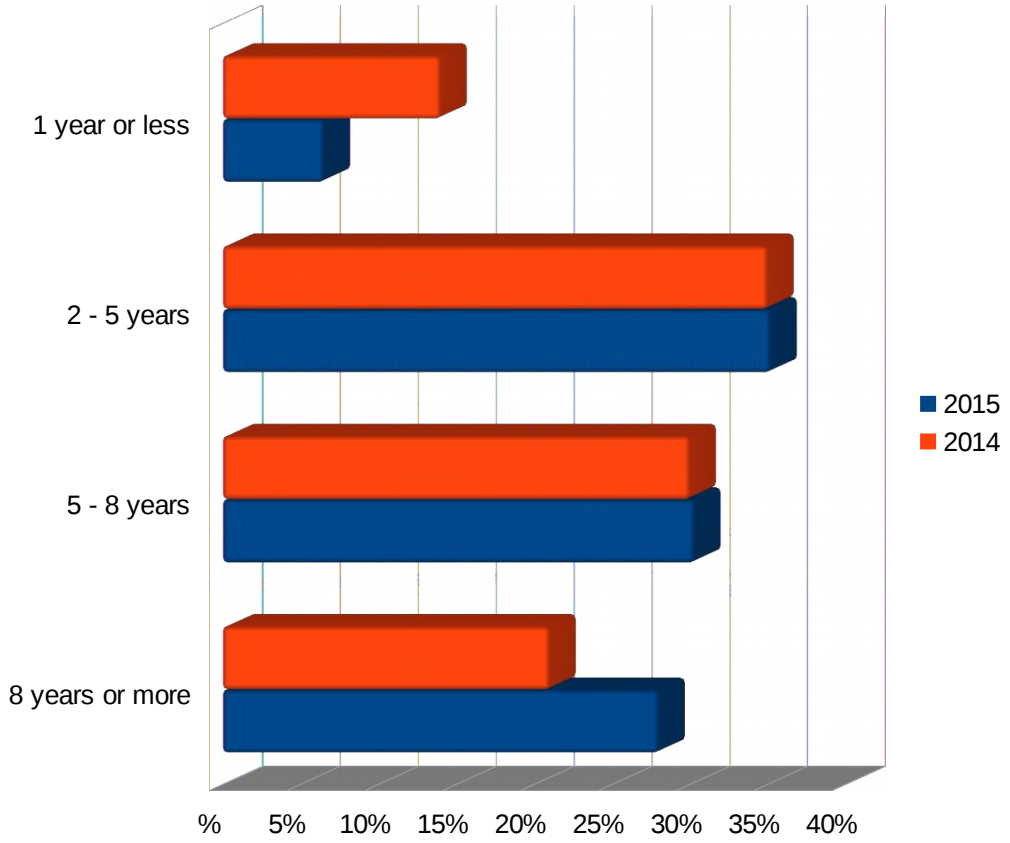
Also notable at least is that all TLS libraries were checked by at least one user. Some of them are rather niche libraries for specific purposes and it wasn't certain we'd reach those users with this poll.

GnuTLS is the unthreatended number two which surprised me a bit since Fedora/Redhat has been shipping curl built to use NSS for quite some time.

I also find it interesting that the OpenSSL fork libressl clocks in at 6%, thus outperforming even NSS and is *way* more used than Google's OpenSSL fork BoringSSL.

How many years have you been using libcurl?

Interestingly, we got very similar distribution this year. What seems to have happened, and this is curious, is that we got about 7 percentage points less on the “1 year or less” category and instead got 7 percentage points more on the 8 years or more one. 28% have been with us since the 7.16.x days or longer!

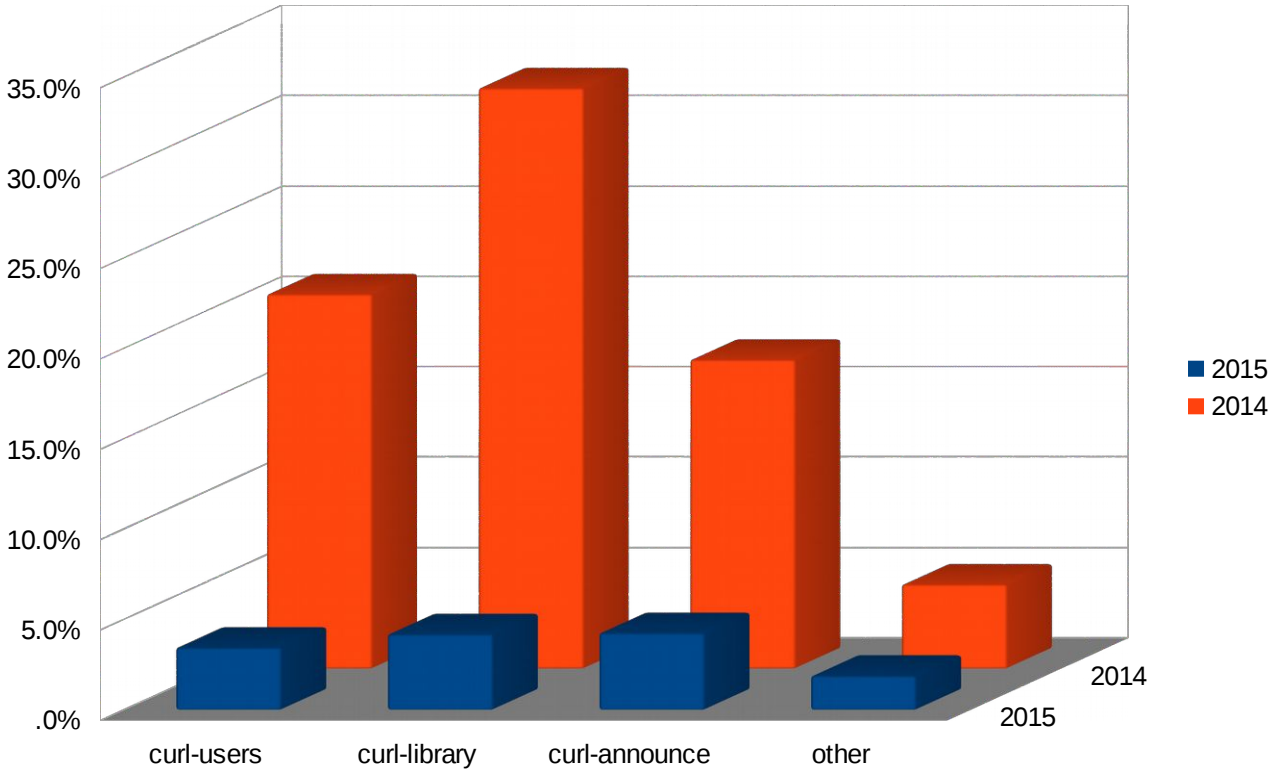


Are you subscribed to a curl mailing list?

Asked mostly to understand who's answering. It shows how good we reach out with the poll outside of our usual ranks.

List	Responders 2014	Responders 2015
curl-users	40	51
curl-library	62	62
curl-announce	33	63
other	9	28

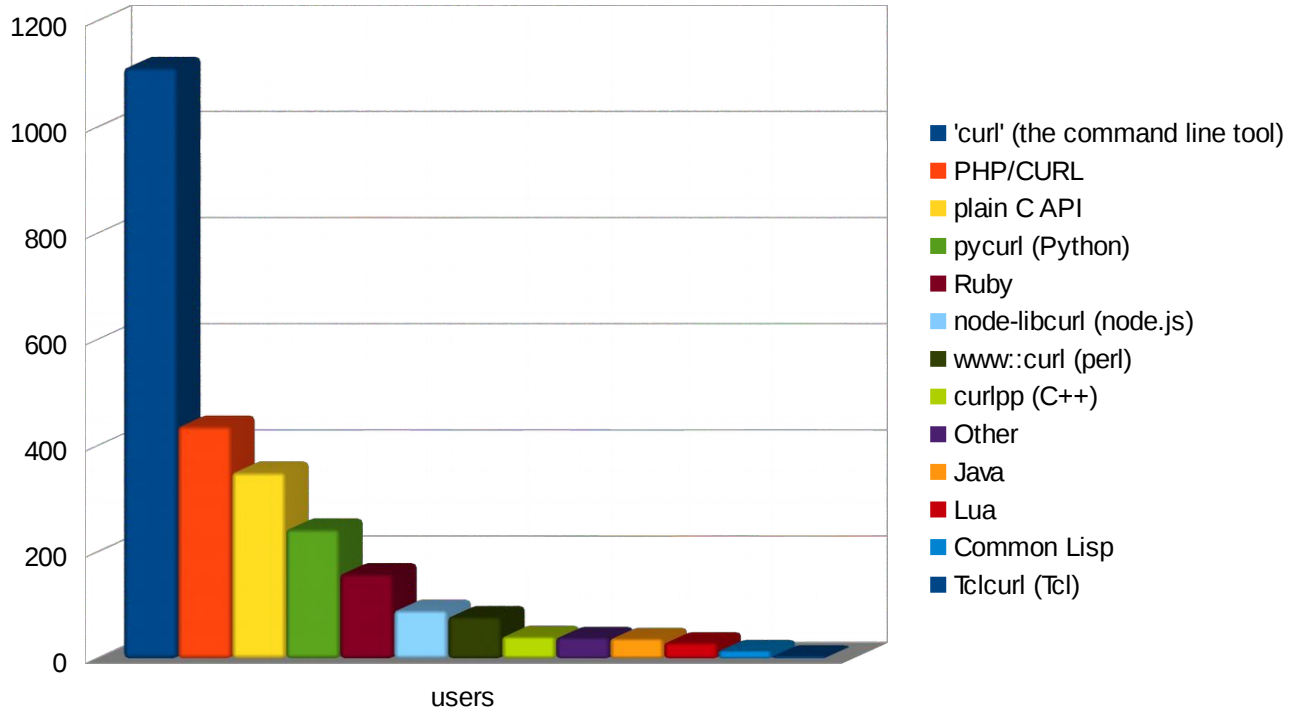
Conclusion: we managed to get a few more of our subscribers to respond. But taken as a whole, it shows that we managed to get a larger number of submissions outside of the ordinary mailing list subscribers. When counting the numbers share among the total number of responders 2014 and 2015 the image looks like this:



Side-note: I don't know what “other” lists people are thinking of when they fill in that. I just imagine that there are other places than the official curl project lists on which people are discussing curl related matters.

If you're using a libcurl binding, tell us which!

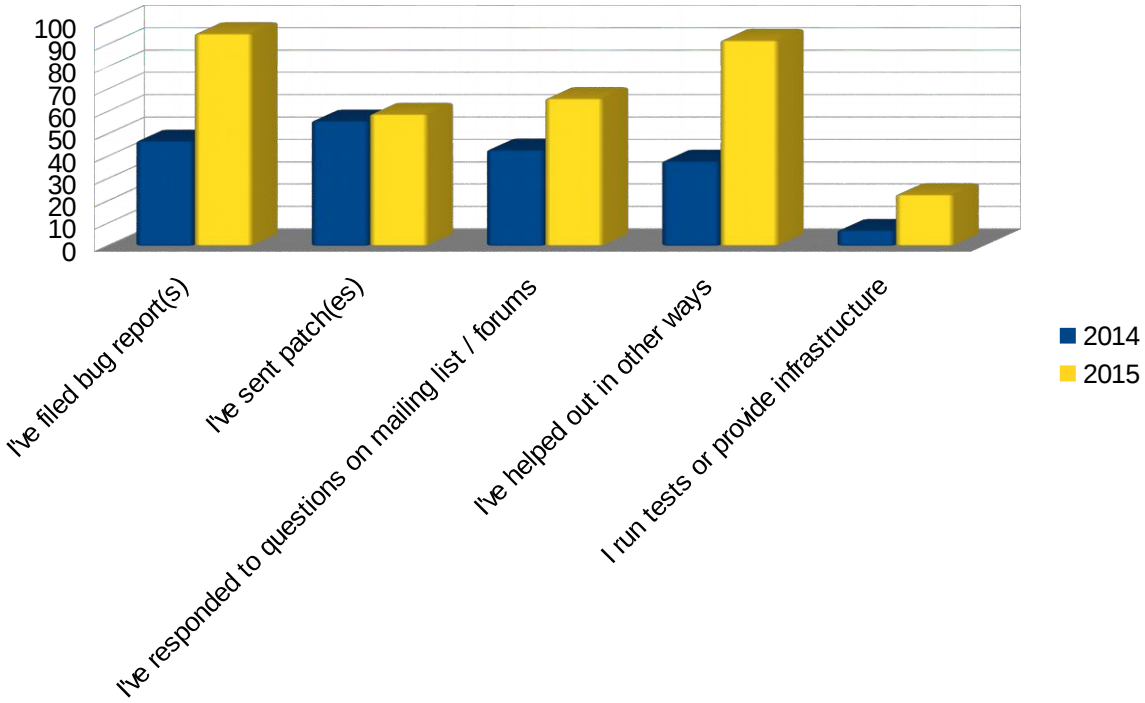
A new question in 2015, to get a feel for the popularity level of the different bindings for libcurl. I'll note here that some people of course checked multiple choices. The “others” choice were mostly other known bindings or custom versions of them.



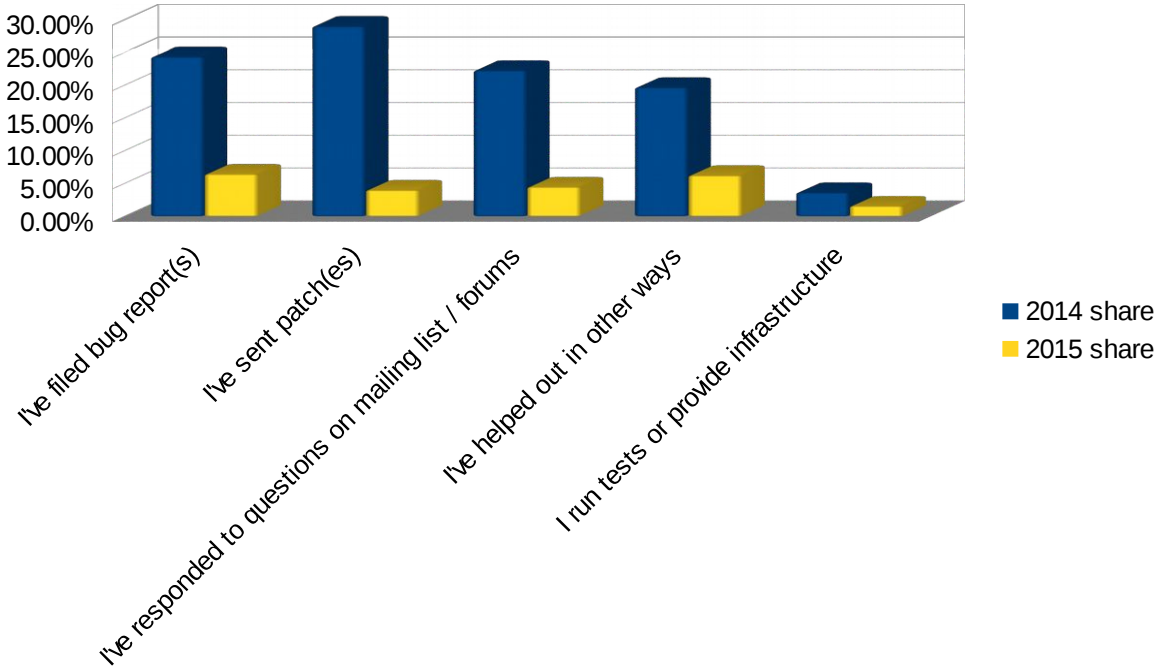
Conclusion: the command line tool is well used and the PHP binding is very popular.

How have you contributed to the curl project?

This truly showed a difference compared to last year. Counting in plain numbers we can see a notable increase in all areas. A total of 216 out of the 1475 submissions (15%) checked at least one choice.



But counted as a share of all the responses, 2015 shows a significantly lower share all over:



Are you involved in other open source projects?

Serves to tell us how experienced our users are with the open source world in general. 2014 we got 78% yes and 22% no. This year the distribution ended up 70% yes and 30% no.

Conclusion: we still have have lots of users not involved in any open source project (other than perhaps curl itself).

What are the primary reasons you haven't contributed or don't contribute more to the project?

We understand that a lot of users don't contribute to the project and that all those who don't have their own reasons for not doing so. This question is an attempt to figure out if there are notable things we do in the project that prevent contributions. Things we should improve.

I think the answers this year again shows that the limitations are really mostly in the other ends:

everything works to my satisfaction	82.3%
I don't have the time	31.2%
I don't know the programming languages used	15.4%
things get fixed fast enough anyway	14.2%
I don't have the energy	13.8%
my work/legal reasons prohibit me	4.3%
Other	3.4%
I don't like or use email	1.6%
I can't deal with the tools (git, make, diff etc)	1.1%
the project doesn't want my changes	0.3%
I find it hard to work with the curl developers	0.2%

A few interesting write-ins for the 'other' field:

“Bad English”, “Don't know where to start”, “curl needs help?”, “don't know what needs to be fixed”, “don't feel competent enough”, “imposter syndrome” (I really cannot figure out who the imposter is), “I don't really have a good excuse”, “not smart enough”, “I don't know enough about the underlying technology”, “I'm not experienced enough”, “no incentive to fix other people's bugs”.

Several people also mentioned the fact that they're using old versions of curl so there “is no point” in fixing bugs in the latest version…

Conclusion: make it easier for newcomers to figure out that we have bugs, how to find them and how to start working on them. Also, maybe add a couple of new alternatives to the set of choices next year.

How good is the project and its members to handle...

Asking the user to grade the project's ability to handle a few different things is just a way for us to figure out which impression we give and how good we are at communicating these things. If we are to believe the numbers, we've basically gotten worse in all areas since last year! Recall here that 1 is the worst, 5 is the best.

In all five subcategories we got less top scores (5) and in all categories we got lower average scores.

I will really have to scrutinize myself and the project to understand why we scored much lower this year, but perhaps the answer is simply that we reach a rather different audience with the questions this time?

During the 12 month period May 2014 to May 2015 we merged patches that were authored by exactly 100 different authors, while we only merged patches from 94 unique authors during the 12 month period before that. We also did 13.5% more commits the more recent 12 month period. Still users think we're doing a worse job this year?

I similarly cannot point out anything we've changed in regards to giving credits or thanks in the project and yet the average score is almost a half point less this year.

The following tables show the answer distribution last year and this year and the deltas between the years.

Patches

	2015 patches	2014 patches		
1	0.2	0	0.2	
2	0	1	-1.0	
3	31	12	19.0	
4	29.4	46	-16.6	
5	39.1	41	-1.9	
Average	4.04	4.27		-0.231

Bug Reports

	2015 bug reports	2014 bug reports		
1	0.4	0	0.4	
2	0.8	2	-1.2	
3	34.8	14	20.8	
4	29.4	37	-7.6	
5	34.6	47	-12.4	
Average	3.95	4.29		-0.341

Minorities

	2015 minorities	2014 minorities		
1	2.8	3	-0.2	
2	2.6	9	-6.4	
3	63.2	43	20.2	
4	12.5	19	-6.5	
5	18.9	26	-7.1	
Average	3.41	3.56		-0.151

Credits

	2015 credits		2014 credits		
1	0.5		0	0.5	
2	0.5		0	0.5	
3	40.5		15	25.5	
4	21		25	-4.0	
5	37.6		59	-21.4	
Average		3.93		4.4	-0.473

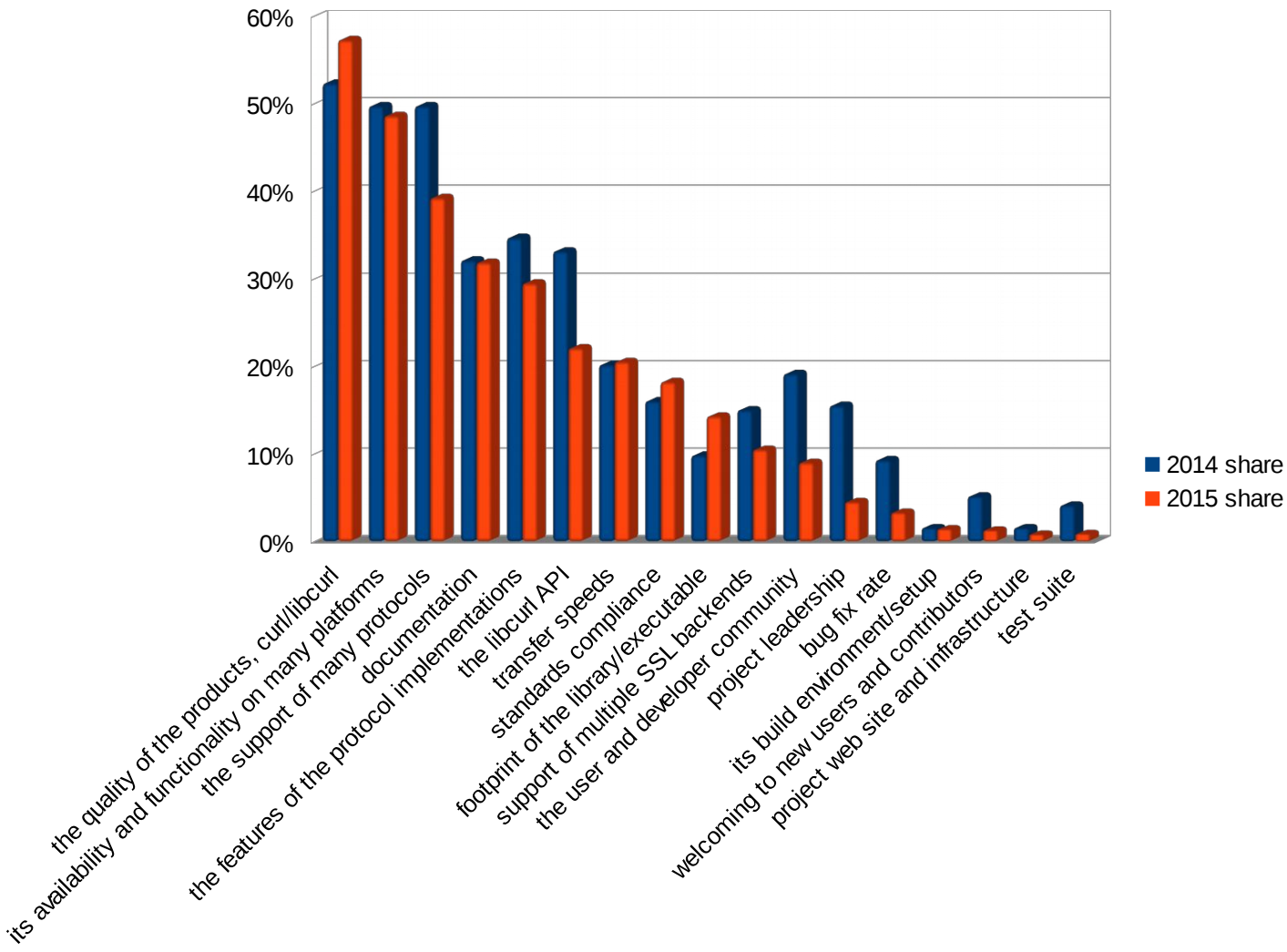
Newcomers

	2015 newcomers	2014 newcomers		
1	0.9	0	0.9	
2	4.1	6	-1.9	
3	48.7	29	19.7	
4	21.5	28	-6.5	
5	24.7	37	-12.3	
Average	3.63	3.96		-0.328

Which are the curl project's best areas?

This year we see the quality of the products race further as the top area for the project. Not a bad thing to be best at if I you ask me. Interesting note here is that ending up forth on the best area top list is "documentation", which is the area that at the same time tops the list of our worst areas. See below for more details on that.

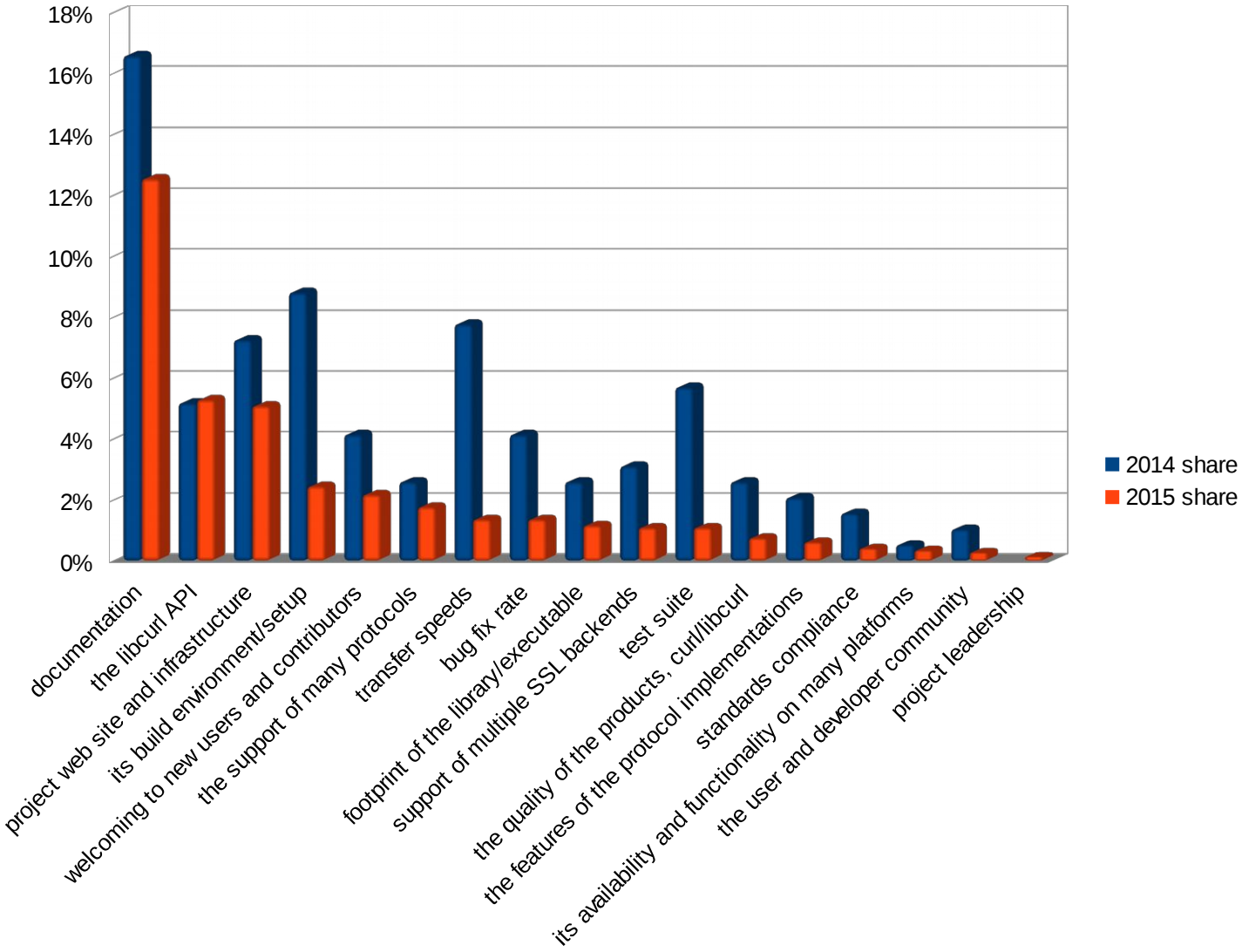
Clearly users have no particular love for our test suite, web site, welcome newcomers, build environment or even bug fix rate. But let's take these answers and compare them to what users said our worst areas are.



Which are the curl project's worst areas?

It is important to note here that users got the *exact* same areas to select from which selecting the best and the worst and they could only pick 5 of each and without ordering them.

Our worst areas should ideally be areas where we should really focus and put emphasis on improving going forward.

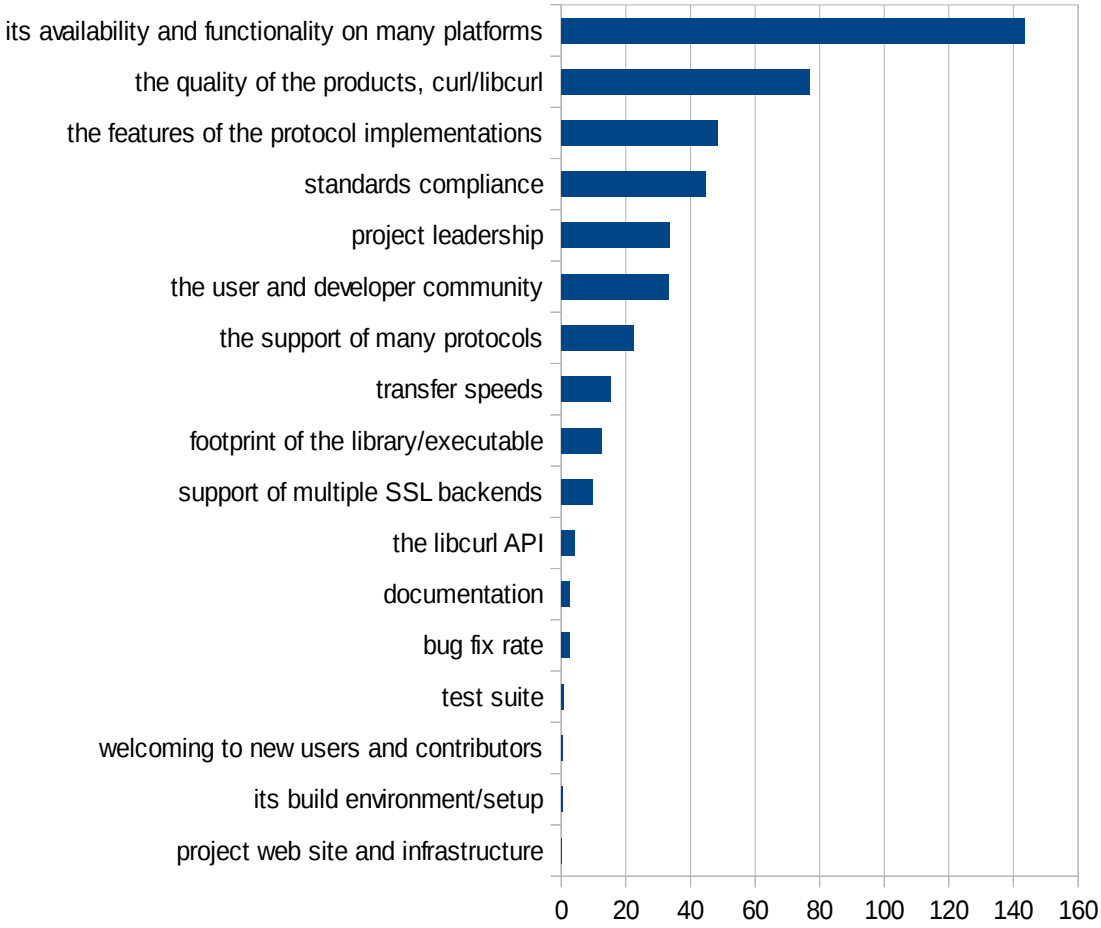


"Documentation" was checked a worst area by 12% of all responders but at the same time was checked best area by 32%! Interestingly, lots of areas got lower shares this year.

Additionally, the fact that documentation got such a high "worst" vote in last year's poll was a primary reason behind the extra effort on improving a lot of the documentation in the project. It doesn't appear to have had a very large effect on user's perception on the documentation.

Best/Worst ratio

Another way to try to understand the results is to count how each area evaluates if we count how many best votes each area got for each worst vote – looking at this year's answers only. A best/worst ratio. The top area then becomes "*its availability and functionality on many platforms*" which got 717 best but only 5 worst, thus scoring a 143 ratio. 143 "best" votes for each "worst" vote.



Web site, *build environment*, *new user welcoming* and *test suite* are the four areas that got more "worst" votes than "best" votes.

Conclusion: we need to improve there.

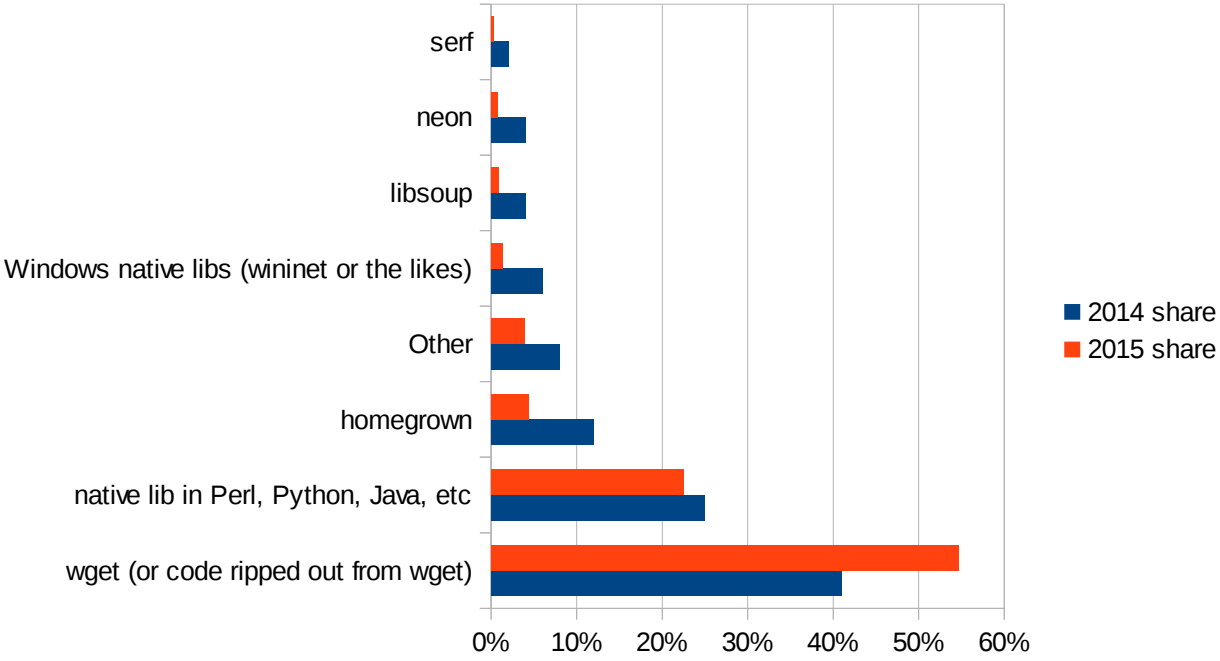
If you couldn't use libcurl, what would be your preferred alternative?

How's the competition? Or perhaps, exactly what is the competition?

Even more now say they would use wget or code from wget while all other alternatives shrunk.

Some interesting write ins for the “other” field:

.NET HttpClient, aria2, axel, C socket library, cpp-net lib, fetch, emacs restclient-mode, httpie, go, Qt, netcat, “No idea – it's too good to consider alternatives”, ruby net::http, “Would Google for an alternative”, “whatever rust http lib ends up turning out best”



If you miss support for any protocol, tell us which!

A completely free text field, so I've had to get all the answers and massage them into some conformance in order to present them properly.

The jokers: HTCPCP (Hyper Text Coffee Pot Control Protocol - RFC 2324) got several mentions but yet not a single one mentioned RFC 1149!

The already done: HTTP/2, SOCKS, SNI, metalink and SSH got mentioned although we already support them.

Ignoring the above, we basically got 32 suggestions and the ones that were suggested more than once were, in order of popularity:

Protocol	Number of suggestions
Bittorrent	5
Websockets	5

rsync	4
QUIC	3
COAP	2

Not a lot new here. These are all protocols mentioned every now and then within the project, even if none of them are any close to getting implemented any time soon – and most of them I would guess never will.

What feature/bug fix would you like to see the project implement next?

This is a little like opening the floodgates. This field got used for all sorts of feedback, complaints, praise and requests. I've manually edited and trimmed this data for readability and brevity.

Output and formatting

- Auto formatting JSON output, Response text styling (html, json, xml), Colored output, JSON decoding. Many mentions of httpie doing this.
- ability to print only the http code and message. a bit like curl -f, but to also print HTTP 200"

Documentation

- Documented and standardized Unicode support in Windows (especially filenames).
- Better documentation. The manpage is accurate, but a list of flags isn't a useful way to learn about something. Can't you sort them by how they're used?
- Cleanup and documentation of the examples, I am never sure if they are up to date or not!
- Improve docs of other than http protocols. Create a man págs forma each protocol, such as man curl.IMAP or curl.LDAP
- Improve libcurl documentation with more examples for each API call.
- Improve the documentation. The curl CLI docs are a flat, unprioritized list of topics, not organized usefully. The libcurl docs are organized the same way, but in practice are easier to navigate (fewer topics to scan through, all topics are ""meatier."")
- Curlpp documentation is also very low quality and takes a lot of searching to even find it, with 404s!
- More friendly documentation - can be really difficult to find/remember the particular flag you need.
- Simpler documentation
- Perhaps break up the --help output a bit to make it easier to find things.

- There is so much on the help list that it takes ages to find the one you need. Maybe put the most commonly needed things at the top, or separate them into sections?"
- Some way to view the usage options per protocol

API

- Better design of the Multi-API
- Make libcurl independent of sockets (similair like OpenSSL but with better API)
- Nicer API for file/binary uploads
- http/https server API
- Setting up libcurl to open parallel connections to multiple IPs for the same hostname is not obvious. This might not necessarily need additional API features, but might be worth documenting. Also, CURLOPT_RESOLVE documentation talks about being able to remove entries (using the ""-HOST:PORT"" syntax), but last I checked, this was not actually implemented.
- Everything works great! I had an issue handling timeouts. It works, but the API a bit clunky.
- I think it would be a good idea to find a way to make the API "idiot proof" with respect to SSL. Perhaps do host validation by default and validate the cert according to the system CA list if the developer doesn't specify otherwise.
- I think that a general API clean up is warranted.
- It would be nice if backwards-incompatible improvements to the API were made in such a way that newly compiled clients got the changes, but clients compiled against older cURL versions kept the same behavior. (There would need to be a "maximum" grace period so that libcurl could eventually shed old cruft, but this would make it easier to improve the API's quirks without making old clients angry.)
- make the url parser api public
- Modernize the C API for HTTP/HTTPS request. Currently even doing simple things like reading headers is much more painful than it needs to be.
- More callback interfaces. A authentication call back would be nice (redirects mean we don't know what the "final" server is so we don't know at the beginning of an operation what credentials to send.
- More complete protocol support (e.g. locking, fancy imap stuff; perhaps curl needs a special directory support api)
- The C API is definitely clunky. Calling a long list of functions to setup parameters makes for funny code. Could do with structs passed in one function call.

Command line usability

- When using ""-K, --config"" with '-' as the filename, it would be great that curl doesn't wait for the EOF from the stdin before ""firing"" the commands
- ability to remove the progress bar
- make curl URLEncode the URL
- A CLI that is less opaque. I spend a lot of time in `man curl` pretty much every time I want to do something beyond "grab this file over HTTPS, write it to stdout/disk".
- a quicker way to discard output than >/dev/null or -o /dev/null, I do a lot of testing and almost never need the actual body. I could write some sort of shell alias or wrapper, but I guess others might find this also useful. Didn't see it in the --help output.
- An option where curl could show me whether my server or my server-side code is doing something bad or incompliant (but that still works). Kind of like a validator. So whenever an error occurs but curl recovers and carries on, highlight that error.
- Borrow httpie's default of not dumping binary data to a tty
- Support for bulk downloads (i.e. making -O work for more than one file)
- Simpler handling of forms
- Currently, if a php xdebug session is enabled, command line curl will just hang. I'd like to see this fixed
- Easier to remember syntax like httpie.
- Given that I am not an expert, I would love to have a GUI to use in order to create a command - so that I can copy/paste it into a batch file
- I think it would be great to have a programmed completionfor the command line (eg. autocomplete on header names etc)
- I'd like one of the '-J', '-L' or '-O' options to support the final file name with 302 redirection, i.e.: curl -v -J -L -O http://www.skype.com/go/getskype-macosx
- run cURL in ""ui"" mode

  $ curl --ui 8080

  navigate to localhost:8080, get a pretty UI for graphically building up requests and nice rendering of responses. Integration with pastebins and the like. Might not be in the spirit of a systems tool like cURL, but it would be handy!
- The UI is less than charming
- shell tab completion for major shells (bash, zsh)
- Simplified command line interface, especially for HTTP / REST

- Turning off the progress meter
- My only gripe is with the "-o" and "-O" options. I keep forgetting which is which! Argh!
- Support for composing simple JSON or query string from the command line, along the lines of -F for multipart form data.

TLS

- implement SSL/TLS for curl homepage. (I know it's planned when letsencrypt.org goes live, maybe it'd be nicer to implement it before them.)
- Ability to embed cacerts in library file, as an aid to deployment
- Being able verify TLS certs using DANE.
- Better support for https client cert in hw tokens (not strictly libcurls fault, pkcs#11 stack/lib quality is just awefull under Linux)
- Easier way to verify self certified certificate even if it means dropping support to some external SSL libraries
- more SSL debug tools
- Certificate pinning under HTTPS
- TLS-PSK
- Unification of different TLS backends debug output.

Protocols

- I would love see ZMQ support
- some way to search all SMB shares on network, like smbtree
- keep going with SMB
- More direct support for CalDav/CardDav protocols.
- SOCKS Proxy Multi-threading support... I have been waiting 5 years for this!
- Websockets

HTTP/2

- HTTP/2 full features. Multiplexing, server push, priorities/dependencies and beyond! [several write-ins]

libcurl

- Better support for asynchronous operations
- Better support for a proactor pattern (instead of reactor as it is right now) (see boost::asio for

more information)

- Improve Transfer performance
- libcurl memory usage should be reduced
- abort function for curl handles
- maybe: common astyle formatting of curl source code
- debugging helpers (like casting the CURL* to whatever it is atm))
- Forced disconnect from another thread
- better support for DNS-based load balancing setups, libcurl seems to be geared primarily toward reusing an existing connection (which is definitely a common scenario).
- Curl appears to prefer its own threaded resolver these days. It would be lovely if it could expose that resolver to client code. We currently use c-ares with both curl and our own code but would probably use curl's threaded resolver instead if we could from our own code.
- I used libcurl for dealing with FTP(FTPS, SFTP) with fopen/fwrite/fread -like interfaces. To implement this I used similar approach, as descibed in examples. On Windows I faced with a bug, working with multi interface. There are race condition on domain name resolving, so libcurl attempts to connect FTP while DNS resolving still in progress. As a result the connection failed. To workaround I reimplemented my libcurl wrapper to use easy interface, instead of multi, and used threads. Will be glad if the issue will be fixed.
- I would like to see curl to detect at run time which ssl library to use, and work with it, instead of requiring curl to be built against a specific ssl library.
- Improve the NSS backend, which has become the standard on RHEL but seems to lack the performance of the OpenSSL backend (haven't yet had the time to look into the reasons, it seems a db locking issue on the NSS .db files).
- control buffer size curl use internally

Misc

- very light proxy autodetection using winhttpd.dll on Windows
- --proto-redir =staysame I'd go as far to say this should be the default for the sake of security.
- minor: use sha256 as a minimum everywhere (f.e. in generated ca-bundle.crt)
- nice to have: test suite to work better on Windows
- header parser
- move development to open repo like github.
- not printing out binary files on the console and garbling the whole shell :)

- recursive http(s) downloads/http mirror
- WARC support
- multiple connections for one file
- support timeout for each easy handle in Multi interface
- provide convenient way to know when the request was return on socket in multi interface.
- A single wrapper for libevent+libcurl as a single-threaded event-pump with concurrent connect/read/write/request queuing and event response dispatching.
- Automated build test suites to avoid regressions like 7.42.0 non-ssl builds axTLS on Windows? Not sure if that's supported
- Callbacks for authentication (e.g., on HTTP 401) rather than callers having to provide them up-front.
- DOM support. Browser like control much like how one could use IE and OLE on windows to remote control the browser.
- Downloading websites and changing the links
- Easier building from source on Android
- Be more explicit to teach the user about wrong usage of curl tool and/or libcurl (some sort of beginner mode)
- Easy to use binary distribution as a static library (!) for Windows, with full SSL support through schannel or otherwise.
- Embedde 'jq' type tool
- Instead of more features I'd like to see you promote current features more so that people notice that they exist. Maybe do some blog posts etc"
- Focus on allowing more build options to (optionally) cut down included features and get executable size as small as possible.
- For debugging purposes, options to have a more telnet-like experience, to receive all the data raw, a bit like -I but for any request type.
- Full CMake support (the right way - without parsing autotools input)
- I've posted a message a long-time ago on the mailing list about adding support for CoAP protocol. http://curl.haxx.se/mail/lib-2013-11/0198.html
- HTML link conversion
- HTTP Chunked Progress! (yes, shameless plug) https://github.com/lloeki/http-chunked-progress/blob/master/draft-lnageleisen-http-chunked-progress-00

- I believe that the way Curl is packaged today and distributed is pretty poor. It is better than other open source projects, but those are abysmal. Once I get the OpenVMS side(s) cleaned up, I'll be in a better position to explain my disapproval.
- I honestly can't think of anything right now.
- I honestly only use curl for http, and I pretty much assumed it didn't have any bugs at all.
- I inherently do not trust CAs, so I would really appreciate a TLS (SSL) certificate pinning option where I could specify exactly which certificate the remote server should have, or optionally also which certificate the remote server should be signed by (in case you roll your own CA).
- I made this https://curlbuilder.com/
- I think focusing on code quality (especially with respect to security) would be the best aspect to focus on.
- Features are nice, but stability and security are paramount.
- I use curl primarily for testing server responses (looking at headers and whatnot). However I want to work on Tor servers and I'd love to see curl support Tor.
- interactive authentication on command line client
- It seems pretty complete already.
- It's always a pain to need a dedicated BitTorrent client when I need to grab something quickly from someone else. If curl supported this, It would likely replace my current use of rtorrent.
- Just wanted to say "thanks for keeping Negotiate/GSSAPI support working", it wasn't mentioned elsewhere in the survey but I use it all the time...
- libuv
- Local cache management! Saving my own headers and parsing out the ETags and Last-Modified dates is a real pain. Yes, I should just use polipo or some other lightweight standalone caching proxy instead.
- Make the CURLOPT_EXPECT_100_TIMEOUT_MS option being set to 0ms the default. It slows everything down and at least I didn't expect this behavior.
- Marketing - get better at telling people what you do
- "Modern C++11/C++14 style wrapper.
- Curlpp is distinctly low quality C++ because of its roll-your-own functors, binding, etc etc. If you are making a small library and it comes with an entire 'utilspp', you know you've made a mistake. Stick to the standard. Stick to common style practices. Do not reinvent

everything, it is detrimental to your users and requires them to learn entirely new ways to do things, new syntax, etc, just for your little library.

- more url pattern matching-stuff
- My biggest issue is the fact that I absolute need the ability to fully control the evaluation of the TLS server certificate, and right now only the OpenSSL back-end gives me that control. The corollary to that is that Red Hat unilaterally chose to break that by switching to the NSS backend, which is not feature equivalent, and that's caused no end of problems for me. That's not really a curl complaint as much as a Red Hat one, but in general the problem is rooted in the fact that the feature set isn't uniform across the back-ends.
- none
- Nothing comes to mind. Just wanted to say thanks for curl!
- "Oauth Auth :)
- OCSP checking of server certificate validity is necessary- what if the certificate has been revoked?
- Pass my SSH public key in a request. For example to communicate with Bitbucket/Github api from the console or Bash scripts without passing my password.
- PHP keepalive across Apache requests
- Please add support for maintaining curl handle to make any SSL session id survive outside of the handle where it was created and used.
- "POSTMan-like UI in the browser
- "Promote the CMakeFile to become a first-class citizen, with all the options
- 32 bit windows builds have issues with downloading a file to the current directory
- make sure cygwin repos have https support in their curl
- Recursive download
- SIGALRM on dns timeout when async is not available. I had to stop using libcurl in a project due to this
- Simplification
- Someone on Hacker News mentioned that redirects can change protocols, which makes sense. https://news.ycombinator.com/item?id=9498180 I only use curl on HTTP and HTTPS, and I would be surprised and would consider it a bug if it allowed HTTP to redirect to an IMAP URL. (The commend suggests a fix of using CURLOPT_REDIR_PROTOCOLS to prevent this behavior.)
- Sorry I haven't done the research, but is there a protocol plug-in format? Or do I have to

build from scratch?

- SSL key pinning
- Client certificates with schannel
- ssl public key pinning like firefox and chrome
- Stop writing to cout/cerr from a friggin library!
- Support for Hawk HTTP authentication protocol
- Support for http pipelining on the command line. It looks like libcurl already supports it.
- --next is awesome. Thanks. I remember, it might have made me repeat some of the command line arguments for each request? Maybe make it so that the options for the previous request can be reused for the following, but will be overridden if specified again?"
- Support for multi step handling of HTTP requests such as form submissions where say CSRF tokens need to be picked up in request #1 needs to be used in form submission in request #2. Think perl-mechanize light.
- Support POST rate limiting. --limit-rate only works for the download/response direction. I've used seen on the server end and used strace to confirm that.
- The command line syntax for sending a file as a POST request defeats standard bash completion (`-data @filename`). I want to send a file more often than I want to send inline data.
- The currently open proxy and NTLM/Negotiate issues
- The one thing I miss is automatic resumption on transient errors. wget does this right, curl should do the same. (Creating a wrapper isn't the right answer, because (a) it's actually curl's job, not that of a wrapper, and (b) it's very hard for the wrapper to work out what actually went wrong whereas it's trivial for curl itself.)
- The syntax is still awkward for saving downloading multiple files simultaneously. I don't want to have to change directory. I want to say: here are the URLS, and just put them HERE in this folder.
- UTF-8 support :(
- windows 10 support
- With so many protocols supported by this point, it maybe makes sense to split curl into multiple binaries with CLI interface specialized for each protocol? They would all belong to the same project and would all get built together but the CLI (and accompanying man pages) for each specific 'sub-curl' would be significantly shorter -> quicker to find the appropriate options for what you want to do.
- Zero-overhead feedback (a way to send bug reports/feedback without an email address)

What feature/bug fix would you like to see the project REMOVE?

The curl project is just as many other projects in the aspect that we tend to keep adding features and functionality over time in a speed and pace that is way faster than in which speed we *remove* things from it.

I wanted to ask people what they think is useless or pointless and we thus can remove. 41 suggestions were made, to compare with the 146 we got for the question of what to work on next.

To no surprise, people here mentioned some of the least popular protocols (see that initial question above), like Gopher, the email protocols, TELNET and LDAP. The by far most common suggestions were variations of “nothing”.

Here are the remaining 16 entries I think deserve being highlighted. Some because they're good, some just because they're odd or fun.

- Automatic collapsing of strings like ../../ in URLs
- big giant warnings about which certificate formats securetransport supports. I know it isn't a curl issue, but curl was the tool which I used and spent much time trying to figure out why a certificate worked on Linux but not on OSX
- C. Do it with D.
- curl have a default way of not verifying ssl certificates
- I dislike that the multiple ssl backends work very differently with client certificates. If they can't support --cert foo.pem then it shouldn't be used.
- I don't like maintaining multiple build systems as they tend to desynchronize (especially VS projects). But that's very opinionated.
- Occasionally I trip over the globbing feature that is disabled with -g. Maybe it should be opt in instead of opt out?
- Please don't remove plain HTTP support.
- progress-bars
- proxies from the 90s
- Remove or make it an option to NOT remove OpenSSL. The semantics of the documentation CHANGE wrt to client certificates so its a cURL thing not a distribution thing. The hoops I have had to jump through to get around that "feature" are so irritating. Make absolutely sure client certificates work in ALL cases/distributions.
- Support for multiple SSL backends. It's important for security to be able to configure secure SSL settings (cipher list, allowed protocols, etc.). There is no reliable way to do this thanks to curl's multiple SSL backends.

- support for old, uncommon platforms
- That retina-burning blue from your webpage!
- The progress indicator being on by default.
- The share API seems to have been subsumed by the multi API.

Which of these API(s) would you use if they were added?

New question for this year. I added it because these are all ideas and suggestions that have been bouncing around the curl community for a long time. I figured it was about time we start to get a feel for what users would think of them. The short summary: there seems to be interest for most of them:

URI handling (parsing/splitting)	520	35.3%
handling of TLS certs/keys in memory	212	14.4%
setting up multithreaded mutexes in an agnostic	99	6.7%

There was also an “other”field that 7 people filled in. The suggestions there have already been covered in other answers.

Should curl join an umbrella project?

No it shouldn't, if our users get to decide. 57% of those who answered said no. 512 “no” vs 392 “yes”, but then there was also the 571 users who didn't fill in an answer to this question.