curl user survey 2018 analysis

“Yed i like it”

summary and analysis by Daniel Stenberg

version 1.1, Jun 12, 2018

About curl

Curl is an open source project that produces the curl tool and the libcurl library. We are a small project, with few maintainers, with little commercial backing and yet we're over 20 years old and we have gathered help from well over 1700 named contributors through the years. Our products run in a vast amount of Internet connected devices, tools and services. curl is one of the world's most widely used software components.

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

Survey Background

We do this user survey annually in an attempt to catch trends and longer running changes in the project, its users and in how curl fits into the wider ecosystem. As usual, we only reach and get responses from a small subset of users who voluntarily decide to fill in the questionnaire and the vast majority of users and curl developers never get to hear about it and never get an opportunity to respond. Self-selected respondents to a survey makes the results hard to interpret and judge.

This should make us ask ourselves: is this what our users think, or is it just the opinions of the subset of users that we happened to reach. We simply have to work with what we have.

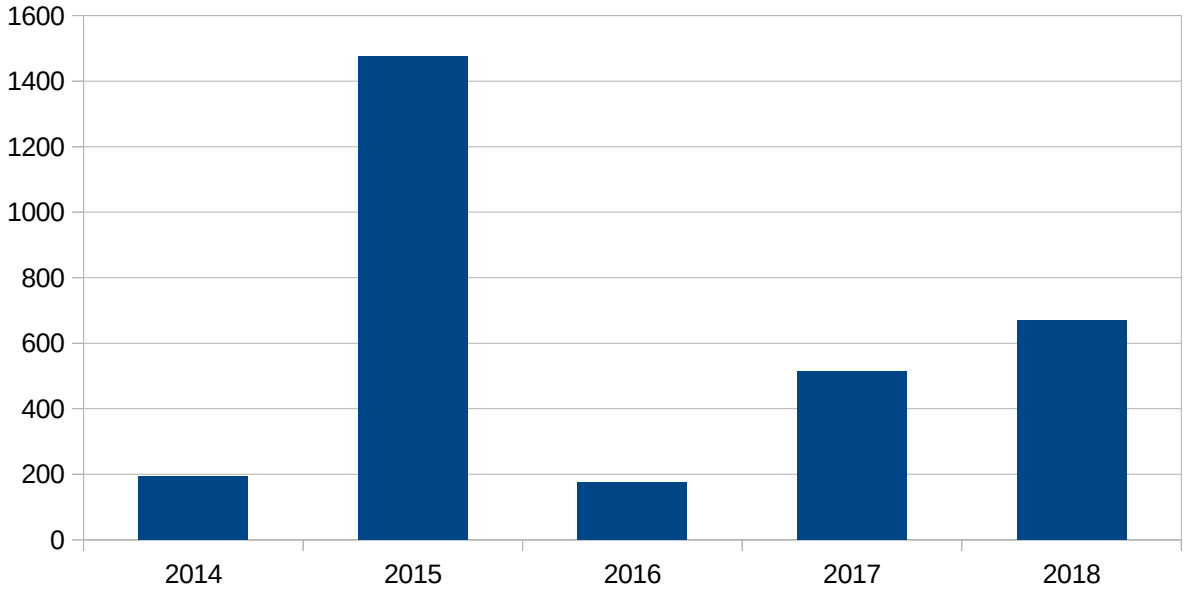
This year, the survey was up 14 days from May 15 to and including May 28.

The survey was announced on the curl-users and curl-library mailing lists (with one reminder), numerous times on Daniel's twitter feed (@bagder) and on Daniel's blog (https://daniel.haxx.se/blog). The survey was also announced widely on the curl web site with an "alert style" banner on most pages on the site that made it hard to miss for visitors.

We used a service run by Google to perform the survey, which typically also leads to us loosing the small share of users who refuse to use services hosted by them. We feel compelled to go with simplicity and convenience of the service rather than trying to please everyone.

Number of responses

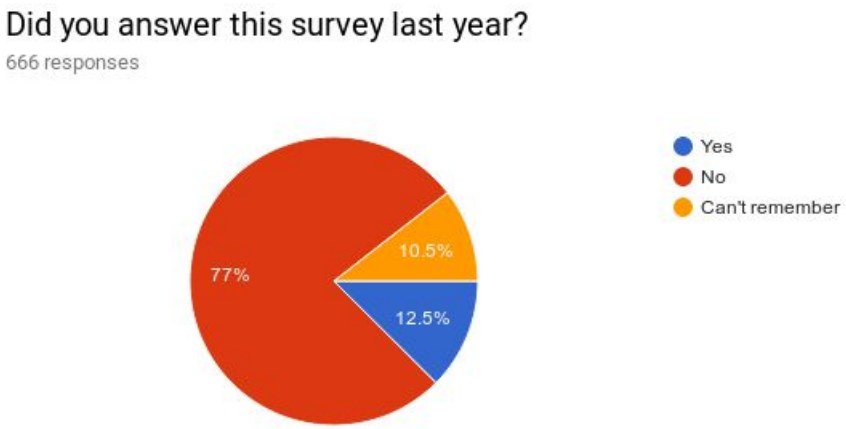
This year showed a 30% increase in number of survey participants compared to last year, climbing from 513 to 670 responses. Still far off from the record year 2015 when a total of 1475 entries were collected.



This year we did however also add some additional questions about the individual answering, to help us understand the responses a little better.

Returning respondents?

If *everyone* who answered this year's survey also answered last year's, would that help us draw better conclusions on how we've changed or not? Maybe, so we asked the question…



so only 12.5% are certain return respondents. 84 persons out of 670. It makes me curious why the other 429 persons who responded last year didn't do it this year...

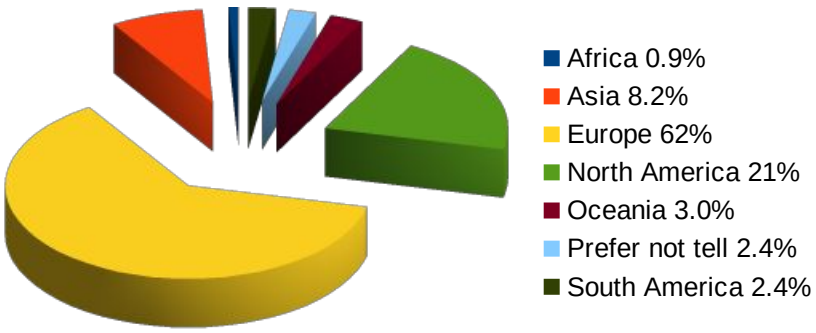
However, trends and changes in the project should however be visible, get felt and be experienced by people independently of their participation or not in this survey last year so maybe it doesn't affect the outcome of all this very much. It is also a good thing that we manage to reach out and get feedback and opinions from new people instead of just recycling the same old.

From where?

Participation in open source projects is something of an expression of wealth. We know the western culture is way over-represented in general but in order to better understand the answers we get in the survey I added this new question this year: “in which continent do you live?”

I decided to stick to a continent level since I wanted just a rough estimate and at the some time not give people the sense that I’m trying to tell them personal details about themselves.

Additionally, I wanted to get this data as input to as how successful we are in absorbing answers from many different corners of the world. To my disappointment, we were even worse at this than I expected. I’m then mostly looking at the number of North Americans, which I originally expected to be roughly on par with the number of Europeans but ended just a third of them.



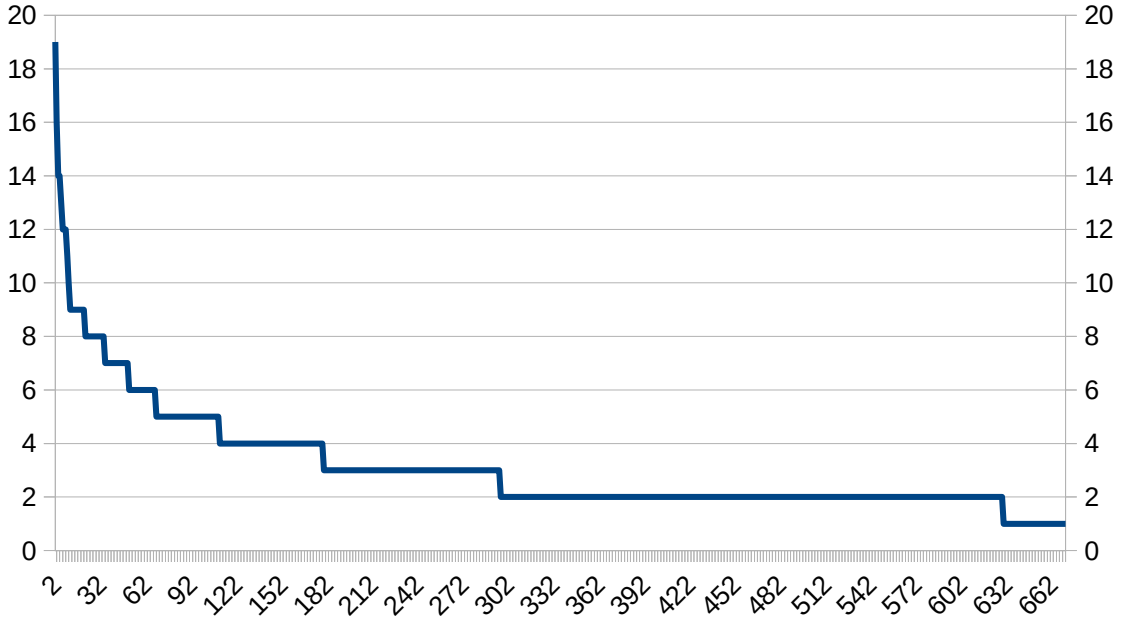
I don’t have any good explanation for why almost two thirds of the respondents are European. I have not gotten the sense that curl has a particular euro-angle among users.

What protocols

n = 667

Possibly the most referred to data in this survey year over year is the distribution of protocol usage by our users. As usual, HTTP and HTTPS are the by far most popular ones as virtually everyone used those, but not even a third of the users used the third most popular protocol FTP (31.9%).

curl is a multi-protocol tool and library. A lot of users use more than a few of the protocols, and this is how the distribution looks, counting **number of different protocols used with curl**. The leading user claims 19 protocols. The average is at 3.12 protocols while the median says 2.

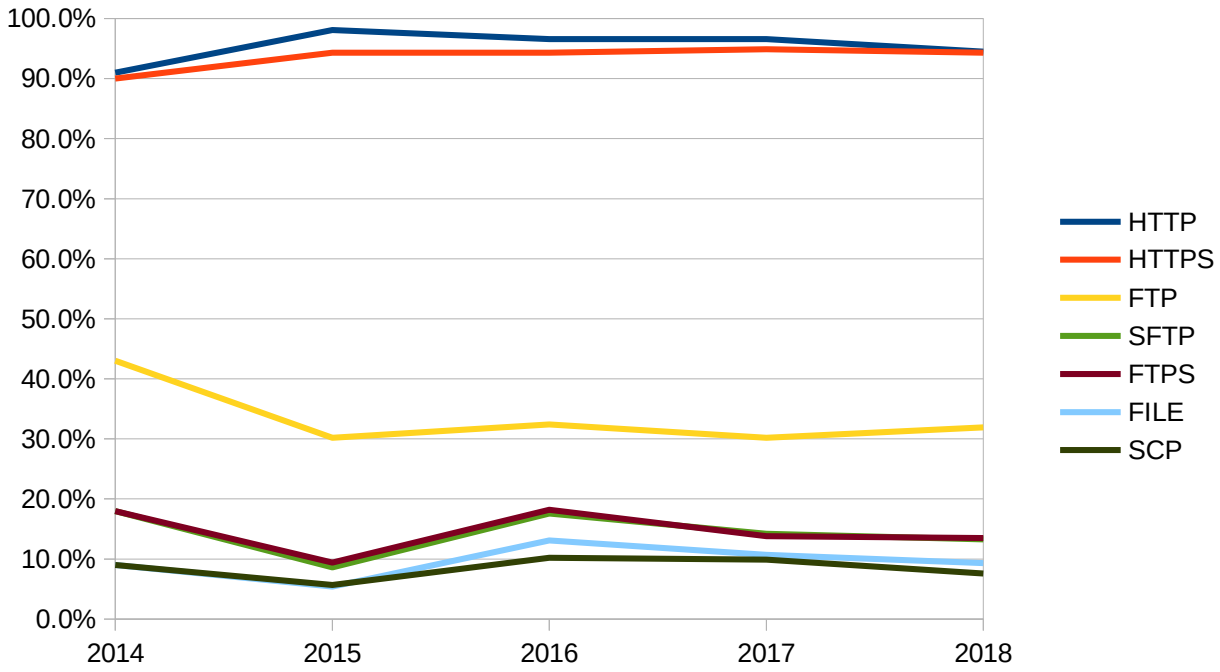


Protocol	2018
HTTP	94.5%
HTTPS	94.3%
FTP	31.9%
FTPS	13.5%
SFTP	13.3%
FILE	9.3%
SCP	7.6%
SMTP	6.6%
SMTPS	4.8%
IMAPS	4.3%
TELNET	4.3%
IMAP	4.0%
TFTP	4.0%
LDAPS	3.7%
SMB	3.4%
LDAP	3.3%
POP3	2.5%
POP3S	2.2%
RTMP	1.9%
RTSP	1.3%
GOPHER	0.9%
SMBS	0.6%
DICT	0.4%

The protocol distribution is *very* similar to previous years - the ones in the very bottom might change individual order from year to year but since they're in the sub 10 respondents that's just noise.

Does the protocol distribution change over the years? Can we see a trend where we have winners and losers? No, not even that…

Looking at the percentage of use among **the top-7 most used protocols in curl over the last five years**, the lines are almost straight… The top seven are the only protocols supported by curl to ever have reached over a 10% claimed usage ratio.

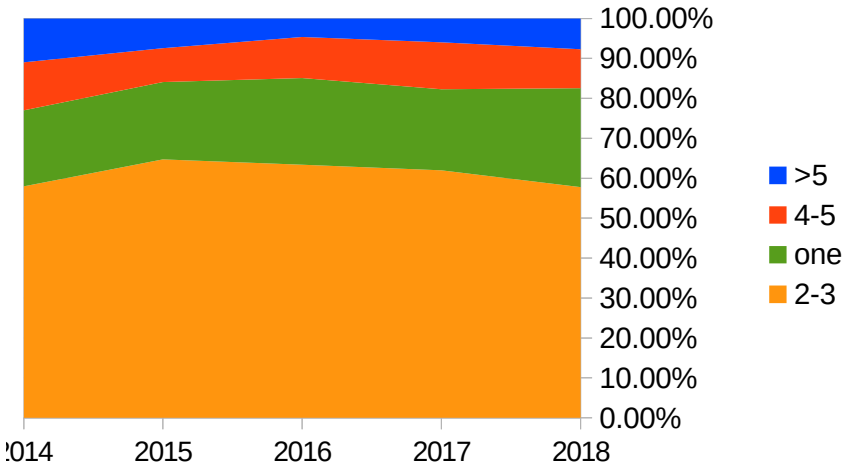


SFTP and FTPS keep being almost glued to each other!

Do you use curl/libcurl on multiple platforms?

n = 666

Just one in four users are using curl on a single platform. All over next to identical distribution as previous years.

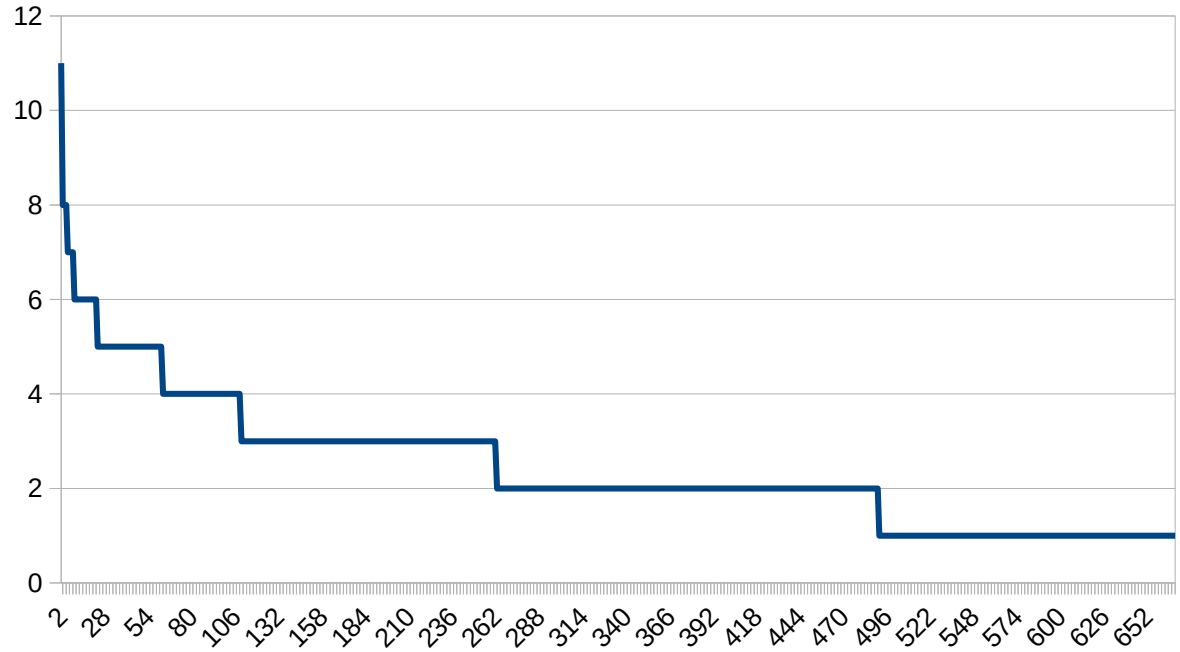


You use curl/libcurl on which platforms?

n = 668

One of the secrets behind curl's success is its availability on many platforms. But do average users stick to using it on one platform or on many? The average user does it on 2.43 platforms, the median on 2 and the most prolific respondent listed 11 different platform running curl.

Number of platforms used to run curl



This question celebrates it's third year in the survey this year, but slightly modified compared to last year so not everything is easily comparable. Now we ask for OpenBSD, FreeBSD and NetBSD specifically instead of just BSD and Solaris was also added.

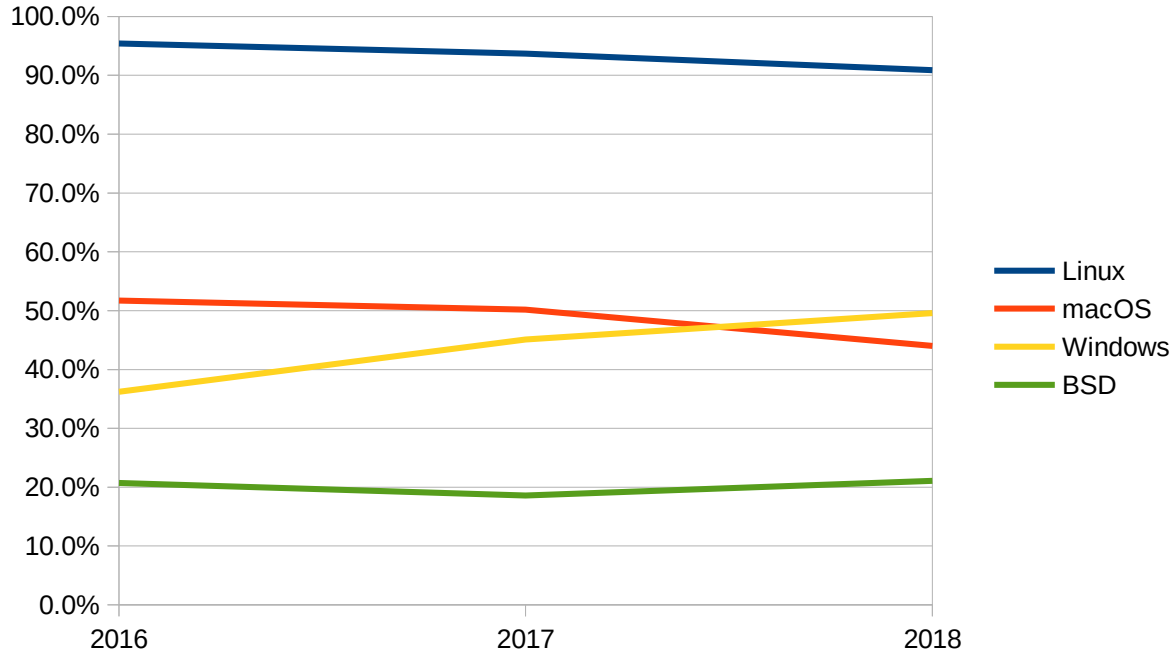
This year confirms the small trend we saw last year when the share of Windows users grew. It grew even more this year and even overtook macOS as the second biggest curl platform, and macOS itself took a notable dive down.

The top-8 platform distribution 2018

#	Platform	User share	Change from 2017
1	Linux	90.9%	-2.8
2	Windows	49.6%	+4.5
3	macOS	44.0%	-6.2
4	Android	16.0%	-0.8
5	FreeBSD	12.0%	
6	iOS	6.4%	-0.1
7	OpenBSD	5.8%	
8	Solaris	4.9%	

The rest of the selected platforms are all at 3% or lower share. VMS being the last in the list at 0.3% but then it seems not a single user selected IRIX this year, down from 0.4% of the users 2017!

Platform usage share changes over recent years. To be able to compare with previous years, this graph has all the BSDs accumulated for the 2018 number. Not a lot of changes...



If you use curl on Windows, which Windows versions?

N = 333

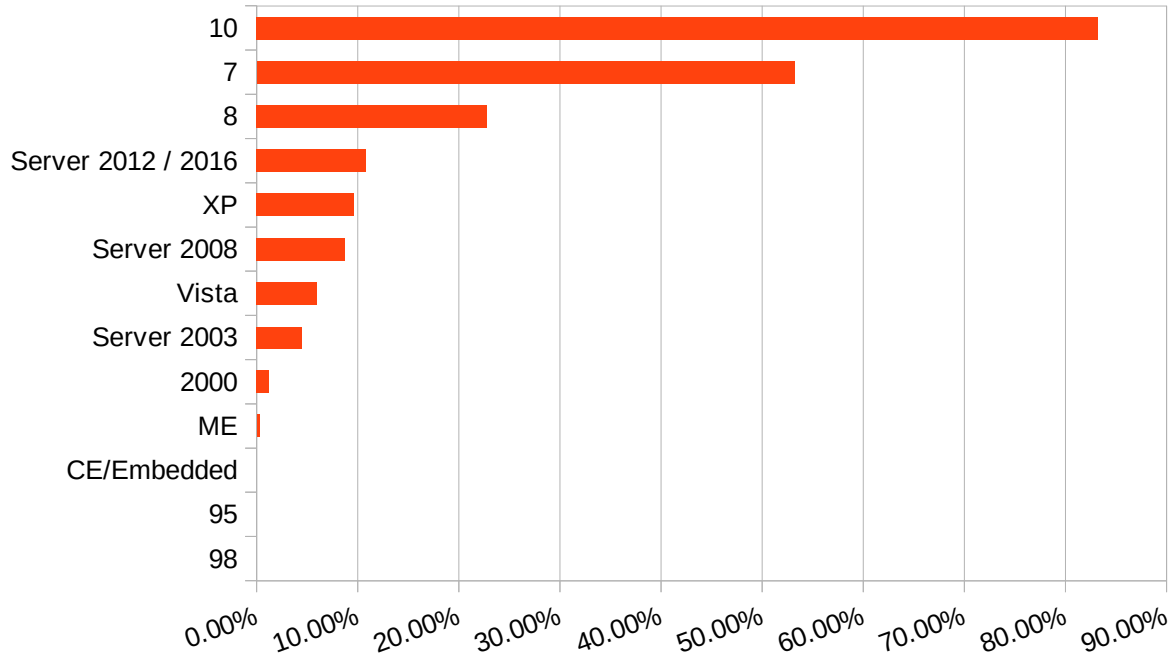
(Interestingly, 333 persons filled in this question while 331 answered Windows on the previous one.)

With basically half the user population on Windows, it is good to get an understanding how the distribution looks like over the many different versions of Windows that are out there.

I was happy to see that at least a vast majority of the users are on Windows 10 (83.2%) or Windows 7 (53.2%), the only versions with more than half of the users, and there was not a single user who checked 95 or 98.

One interesting observation I think is Windows XP which clocked in at 9.6%. Interesting because it is often referred to as “deprecated” and “unsupported”. Clearly there are still some XP installations out there running curl…

Amount of Windows users who stated they're using these windows versions:



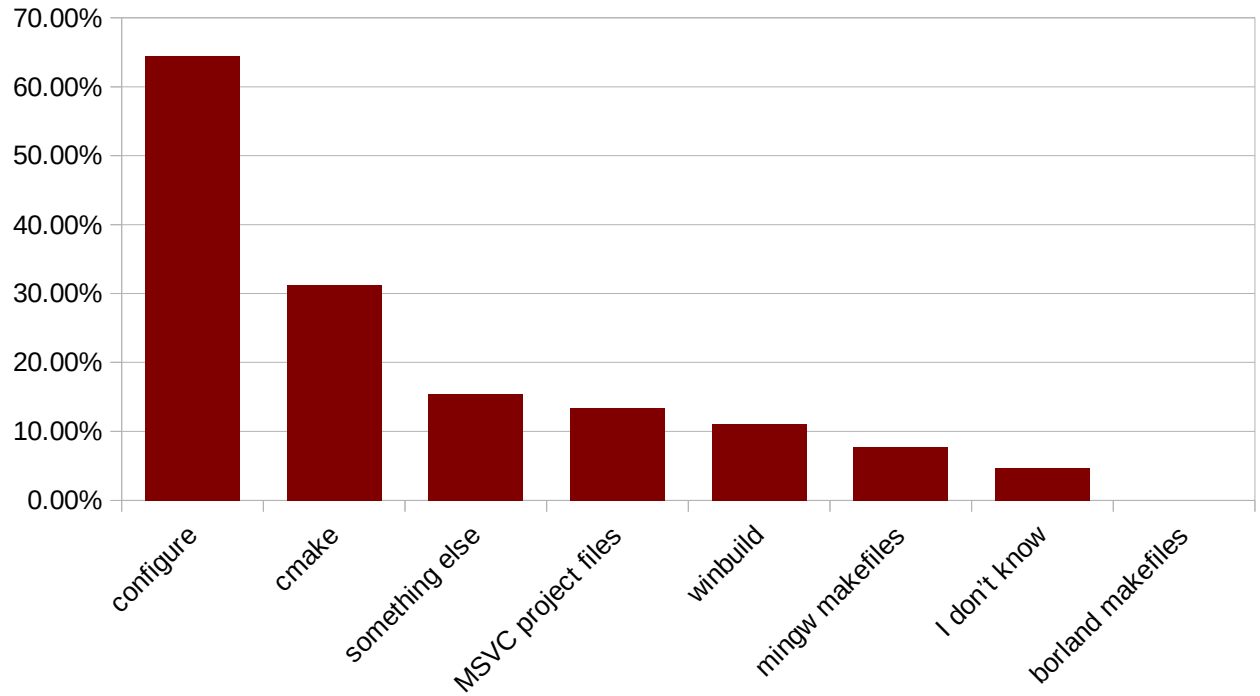
Do you typically build curl/libcurl yourself?

n = 657

This is my attempt to get a feel for what build system that are most used and thus most important to existing users. Of course 70% who answered this question said they don't build themselves (up from 66.9% last year), so I'll focus on views of the remaining 30. The bars below are as a share among those who didn't answer "no".

This year the question listed all build options we provide, and it also became a multi-choice answer so it isn't easily comparable to the results last year where users could only pick one answer.

Share of users who use what build system to build curl/libcurl



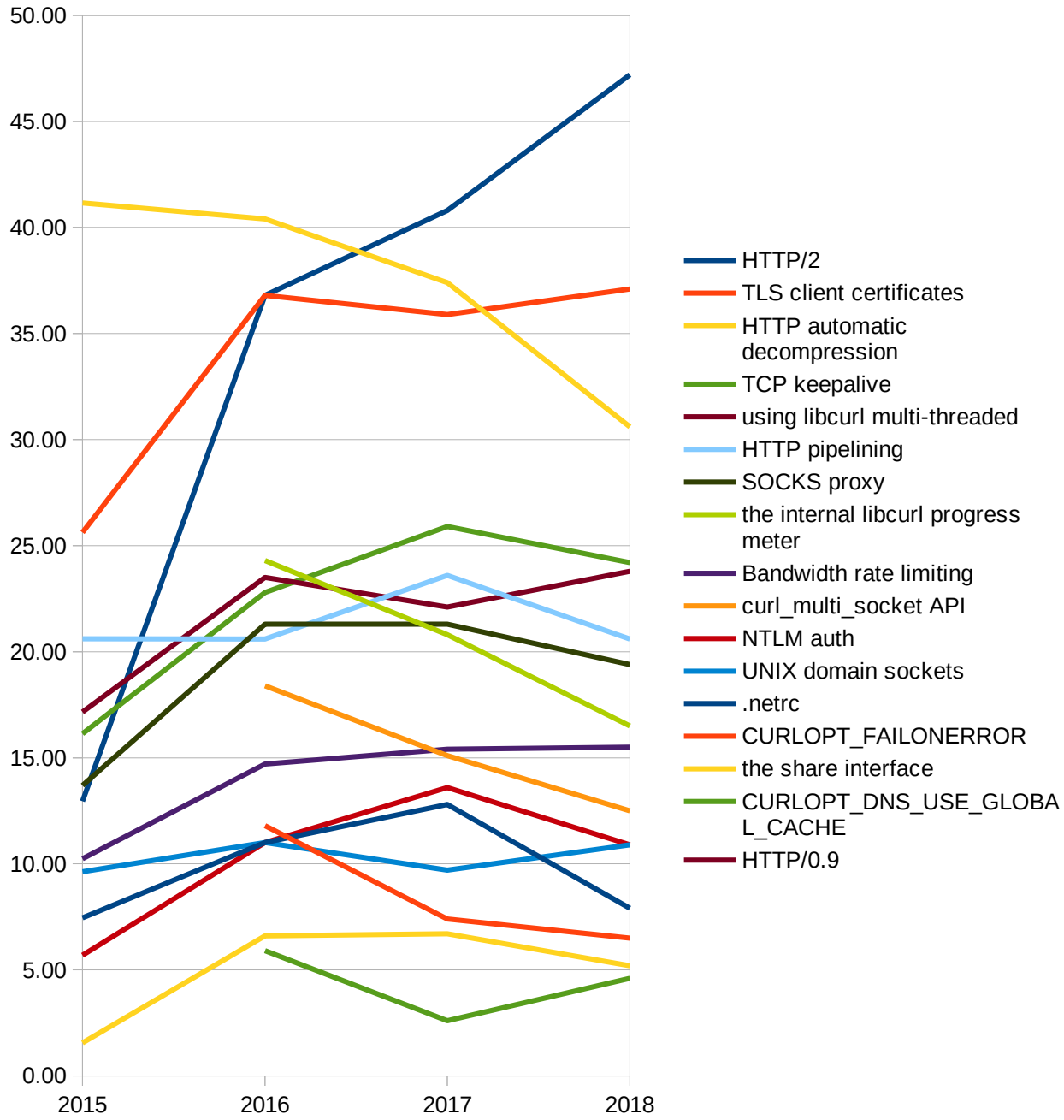
The *zero* users of the Borland makefiles will make me move on removing them from the source tree in the name of simplification. (The subject has already been brought to the mailing list and a PR has been filed.)

Tell us which libcurl features you use?

n = 504

This is a personal favorite. With the answers here we can really see what's important to libcurl users and what features that we could start discussing deprecating in a future.

This year I added HTTP/0.9 to the answers since we've been discussing maybe dropping support for it. A 3.8% usage share, I'd say is not insignificant. Most of the features remain at a similar usage share as previous years as you can see here...

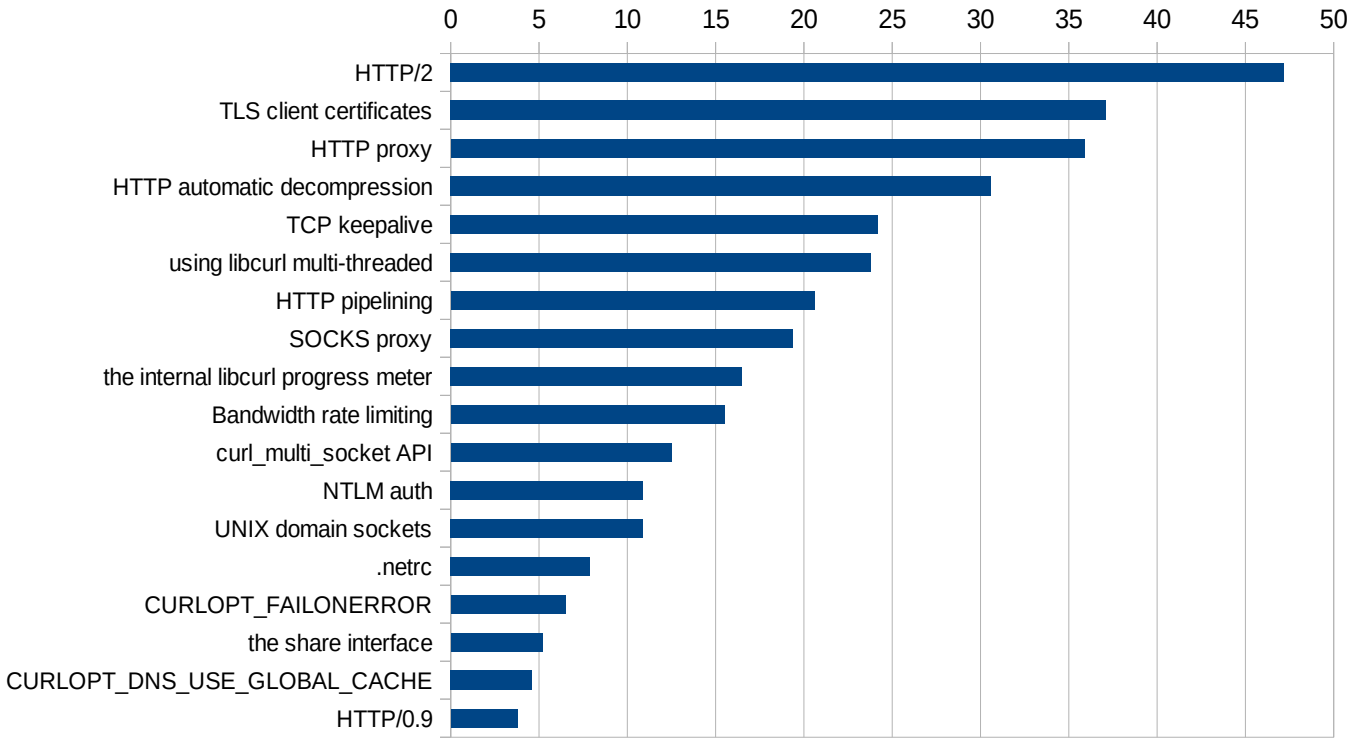


Obvious trends: HTTP/2 is really still booming. Now used by almost half of the users population. That's a 3x times increase on three years and an almost 16% increase since 2017.

The downward trend for automatic compression is hard to explain, especially since libcurl during the last year also got support for brotli which allows for even better compression ratios possible making the speed gain higher.

The usual caveats go out for the client certificate number which continues to be much higher than I suspect is "real" (and continues to grow too!) as I continue to suspect that a share of users can't keep all the different certificates apart.

Personally I think the share interface is a little bit of a hidden gem in the libcurl API family and I consider it unfortunate that it doesn't get more traction than 5.2% now.



Which SSL backends do you typically use?

n = 638

There was a new answer in this year's version of the question and no less than 18.3% of the users selected it: "I don't know". I have no problems with that as I actually think most users don't *need* to know what TLS library their curl build is using. However, I think this skewed the numbers a little since previous years the "I don't know" users would just have skipped the question entirely instead. I think this may be the biggest explanation to why several TLS backends show a falling trend.

This is a multiple-choice question (since many users run many different versions of curl) the sum is way more than 100%.

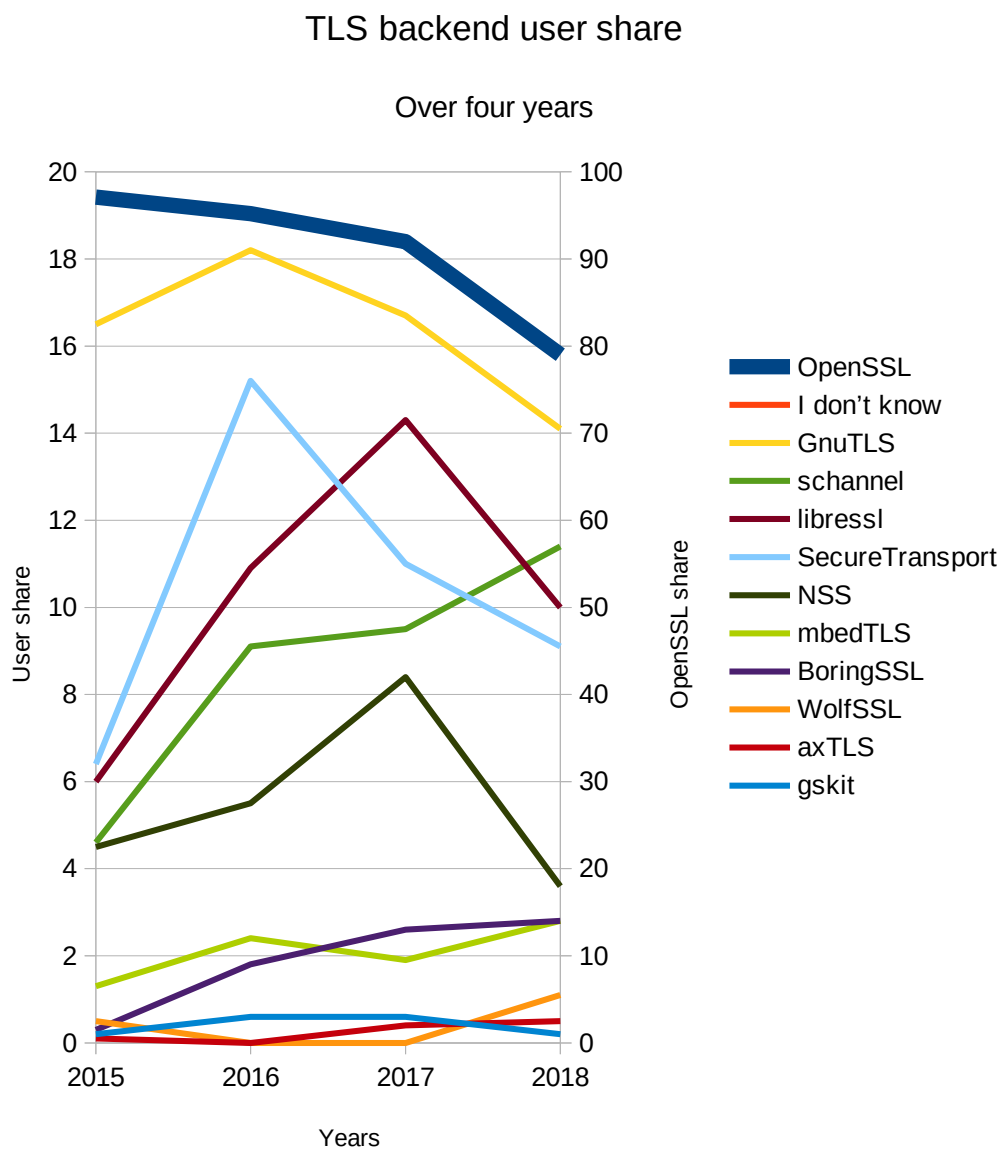
2018 shows that the last year was no anomaly. The OpenSSL share has decreased. It was always the by far most used backend and it certainly still is – without any real competition – but this is the first time where a fifth of the respondents don't use an OpenSSL powered curl. **Note that in the graph,**

the OpenSSL line uses the right-side Y-axis while all the others are on the left-side Y-axis. Done so to make the lesser used libraries' graphs easier to view together in the same graph.

SecureTransport's decreased usage could be attributed to the fact that Apple dropped it from their curl build in favor of libressl, but that makes it harder to explain why libressl also took a significant dive! Redhat has switched their curl builds away on Fedora from NSS to OpenSSL and that can explain the NSS fall. Schannel is on the rise for the third year in a row. While Microsoft now ships a native curl in Windows 10 built against schannel, that is still such relative new development so I think that's not actually the full explanation… I think it simple has taken time for people to switch over and discover that this backend still works pretty well while removing an external rather large dependency.

BoringSSL keeps climbing this year and is at 2.8% share larger than it has ever been among curl users. This year mbedTLS clocks in at the exact same share as BoringSSL. WolfSSL gets a significant bump as it 1.1% now.

At 0.5% the axTLS user share is a dying breed as this TLS backend is now being deprecated and deemed unsuitable for use with curl (for quality purposes).



How many years have you been using curl?

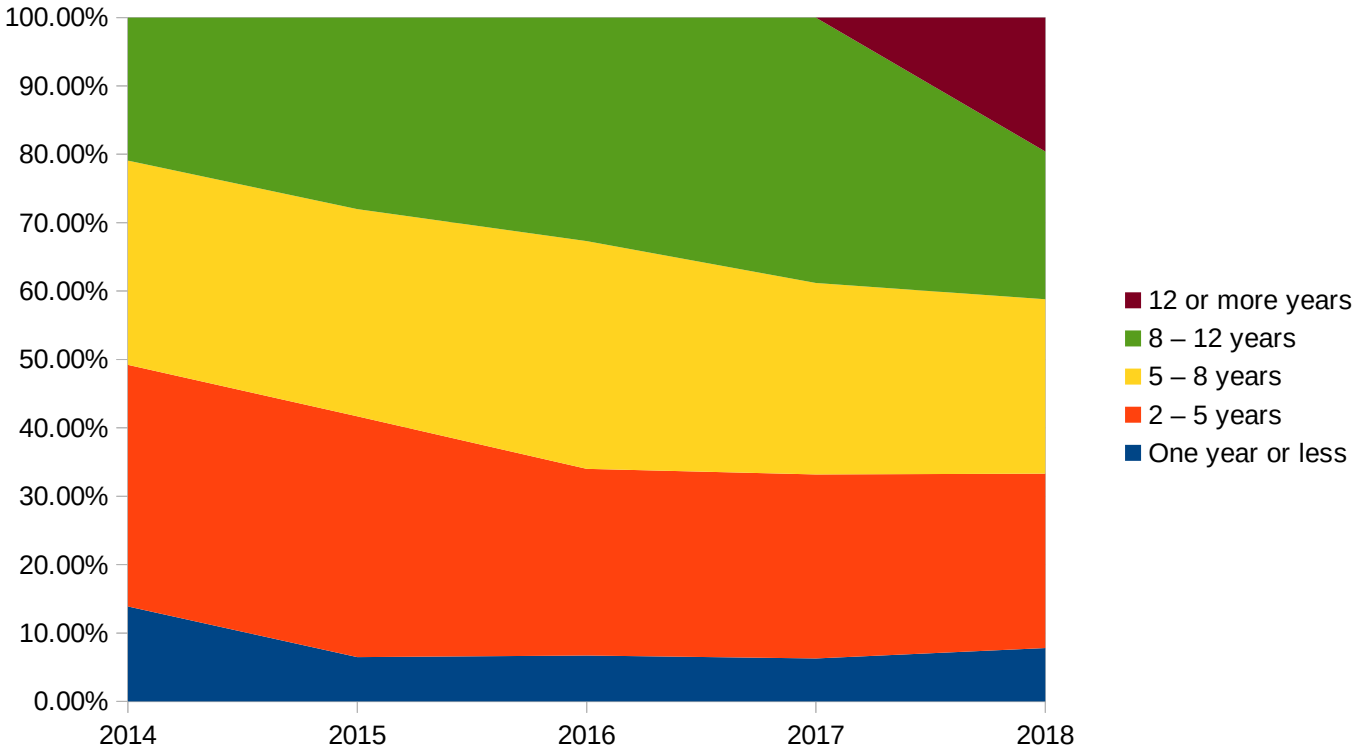
n = 663

Having an old user base could be a sign that our products are trusted and can be relied upon for a long time. But also having a decent share of newer users is a sign that we still are relevant for newcomers into internet transfer tools and not only a choice for an older generation of users. I think it might signal good health.

One in every 5 curl users have used it for twelve years or more. Pretty impressive I think!

The “curl age” distribution is remarkably stable over time. This year I added a new upper slot among the answers. Previously the oldest used to be “8 years or more”, but now that become “8 -12 years” and there’s a new “12 years or more” answer.

So, with that in mind, the green field from previous years is now split up in two, but those two fields together only grew a little bit compared to the single green in 2017. The one year or less category when up from 6.3% to 7.8% in this last survey.

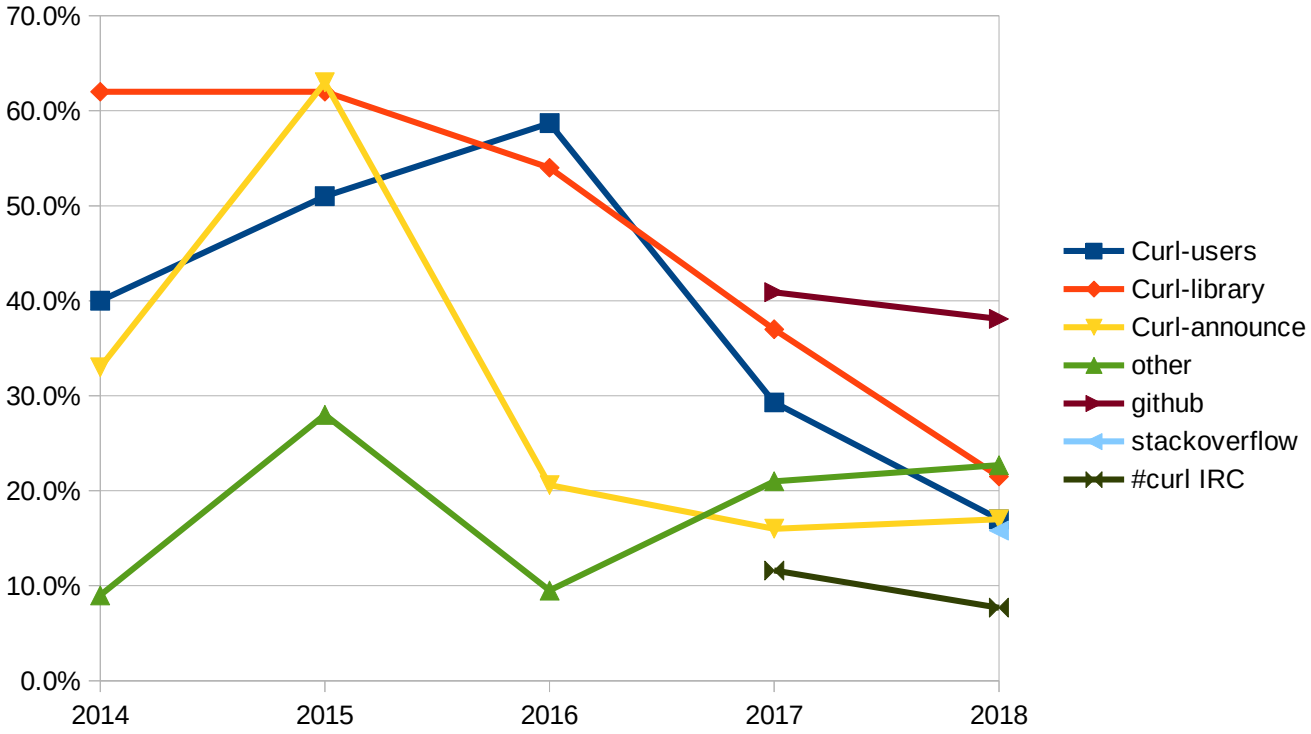


Channels you participate in

n = 247

First we should note that only 37% of the population answered this question that already is very hard to draw any conclusions from. In general less and less of the respondents check alternatives so they’re all falling year to year. Except for the magic “other”, which could be a signal that we’re getting activities done more distributed in more areas as we grow older.

Stackoverflow was a newly added alternative this year.



How do you access libcurl?

n = 596

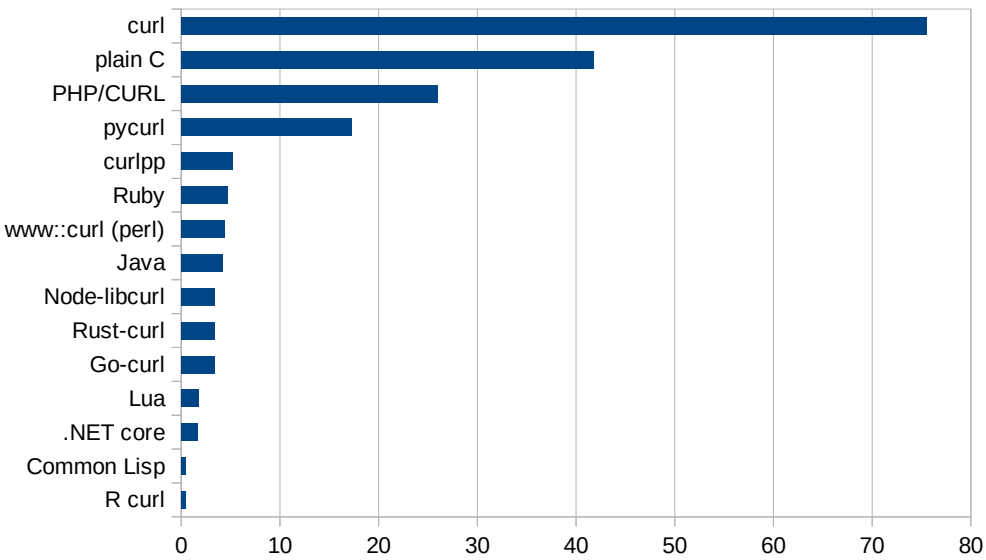
“access” in the question means how you run libcurl. From the command line tool or what binding/API.

Clearly the command line tool and the C API are the two primary drivers, with the PHP and the Python bindings being the two dominant bindings.

This year ‘tclcurl’ was removed from the selection of answers (and not a single user wrote it in). Newcomers are the rust, go and node bindings and interestingly they were all selected by 3.4% of the respondents.

My favorite write-in for this question was “forth”!

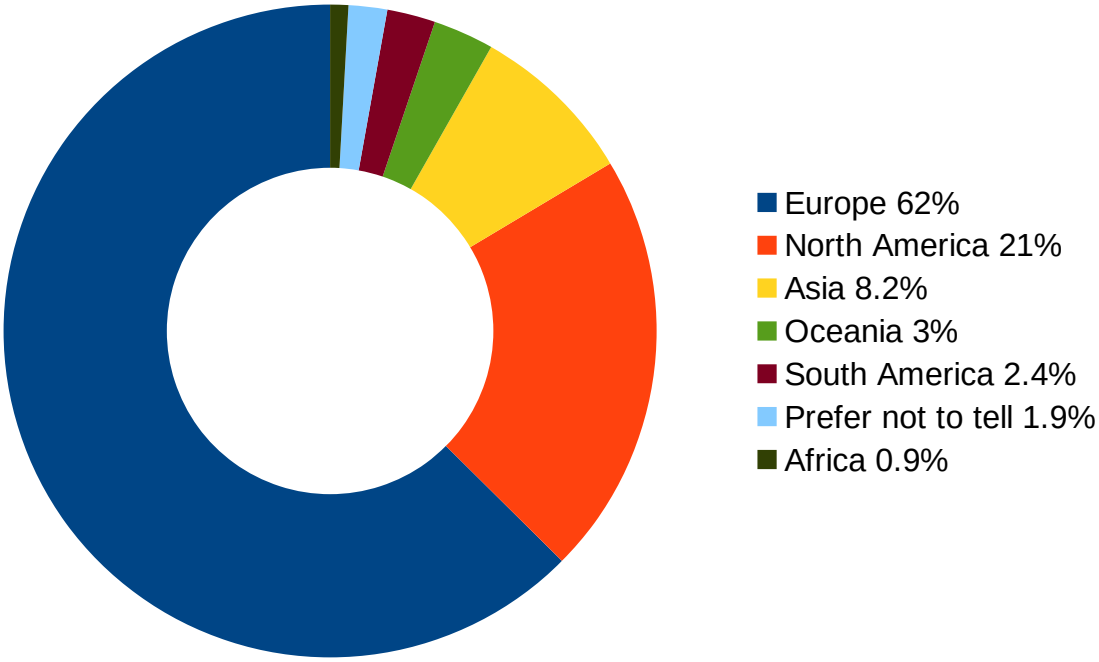
The graph shows the percentage of users who selected which...



Where do you live?

n = 667

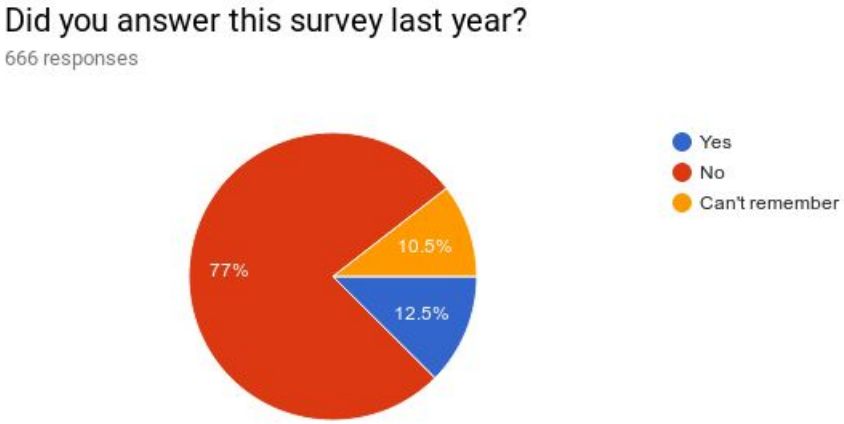
This question has never been asked before. I was a bit surprised about the strong European dominance among the respondents. I expected North America to be roughly on par but instead it ended just a third of the Europeans. Africa was probably unsurprisingly the continent with the lowest number of respondents at 0.9%.



Did you answer this survey last year?

n = 666

Another new question. Meant to serve as a helper to judge what changes in the survey means as compared to previous years. Turns out the vast majority of respondents did not participate last year and more than a tenth can't remember if they did!

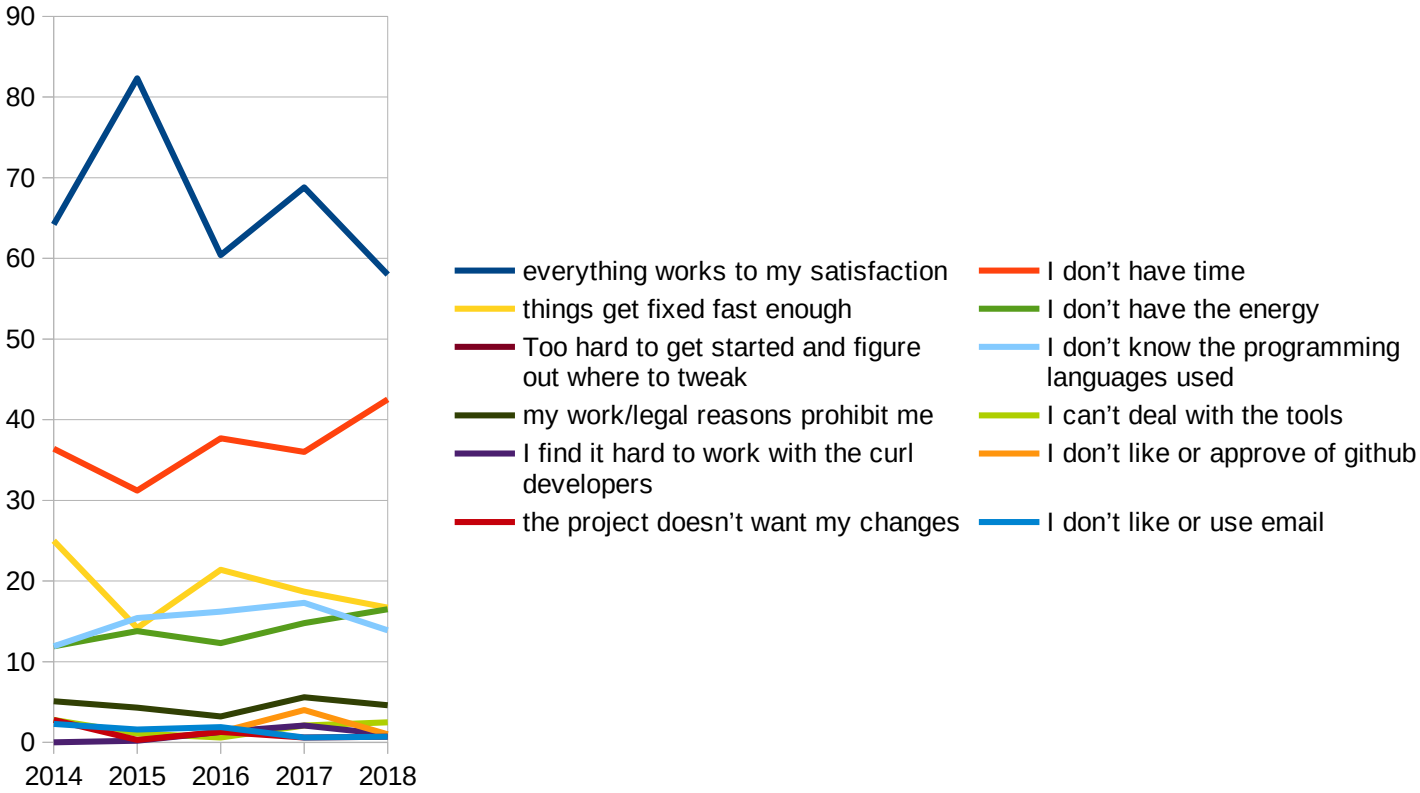


What are the primary reasons you haven't contributed or don't contribute more to the project?

n = 612

I want this question to guide us what we can do better to make sure everyone who wants to can contribute. It could possibly also work as a guide in how we improve or get worse on these matters over time. In reality it seems the distribution of the answers are roughly the same, year after year. “everything works to my satisfaction” is at 58% still the top choice while the “I don’t have time” answer is at 42.5% on an all time high.

“Three children” might be my favorite write-in here.



What could the curl project do/change to get (more) contributions from you?

n = 79

The companion question to the previous one. This is entirely free text so I find it encouraging that almost 12% filled in something. To present the answers sensibly, I’m summarizing them here to my best ability.

1. Enhance documentation

Example response: *A well written beginners guide. Who just studied Python basics. Don't know where to start.*

I spend a lot of time on writing docs and improving docs. Not too long ago we introduced the “help us” web page as an intro to the project and how you can start helping out: https://curl.haxx.se/docs/help-us.html

I would be thrilled to get feedback on how that document works and what we need to do with it to make it better.

2. mark “low hanging fruit”

Example response: *advertise low hanging fruit and how to help*

I know a few other projects do this, but I honestly can’t see how we can make this work. The easy bugs are typically the ones that get fixed almost instantly. To make such a system work, we would need to deliberately keep bugs open just to allow newcomers to have easy bugs to fix. Keeping bugs open on purpose doesn’t sound like a winning concept for me.

We already mark bugs with “help needed”. Those are bugs that we have confirmed are bugs but are currently not able to work on in the near term. Excellent candidates for someone who wants to help out.

3. Accept PRs faster (even in feature freeze periods)

Example response: *Accept a large patch for review even if it's not that part of the development cycle*

We *do* review PRs even during the feature freeze period even if we don’t merge them until the feature window opens again. We’ve chosen to work like this simply because we don’t have people enough and infrastructure setup to do a lot of parallel developing in multiple branches so we stick to a single one.

And besides, we have the feature freeze periods to encourage people to help out with fixing and merging bugs and by insisting that we instead focus on future new features, we put less emphasis on the bug fixing part and that’s not really a good long-term model either!

When we get more maintainers involved in the daily development, we can reconsider adding a next-release branch for such merges.

3. Add extra hours to the day

Example response: *Globally introduce 72 hour days.*

There were a lot of variations of this one. Humorous ways of saying that they don’t have time. I fully respect that, but unfortunately that’s not something we in the curl project have figured out how to fix!

4. Leave some bugs for us!

Example response: *Be less awesome*

A lot of responses stated that there aren’t any bugs (affecting them) to fix and that we’re fixing the few that do affect users fast enough.

5. Nothing!

Twelve something users said there’s nothing we can do. Kind of futile way of looking at the dilemma, but I suppose it is actually another way of saying that the problem is not in our end but in lack of time or ability etc.

6. Be better!

Answers that didn’t really fit one of the five categories above. I’ve collected them here verbatim and without a lot of comments. Here’s what respondents suggest we should do in the project to make it easier for people to contribute. It is hard to understand how adding new features etc could make it easier, but hey, I’m not judging here...

- build system backwards compatibility
- Decrease bug reproduction hassles (e.g. setting up a server, configuring SSL backends...)
- DNS resolutions using SRV records
- do you have webshop to buy something? We need invoice for donating.
- Drop support for Windows < 2000 and/or GCC < 2.95 as there's no way to test this and no documentation available anymore.
- I could run tests if there is a simple way to run them, e.g. via cron.
- Preferable language and community
- Re-implement curl in Rust
- Sell me curl merch that contributes to development
- smaller, more focused libraries are better than behemoths. Curl is too big.
- Stuff gets unreasonably denied, like -f variant that outputs response error status text and body.
- Support DANE (DNSSEC based TLSA records)
- To get more docs on PHP cURL
- Try to drop off support for old platforms and less used features.
- Yed i like it

That last suggestion made me giggle. Sure it is certainly a typo of some kind but I still can’t figure out what it means…

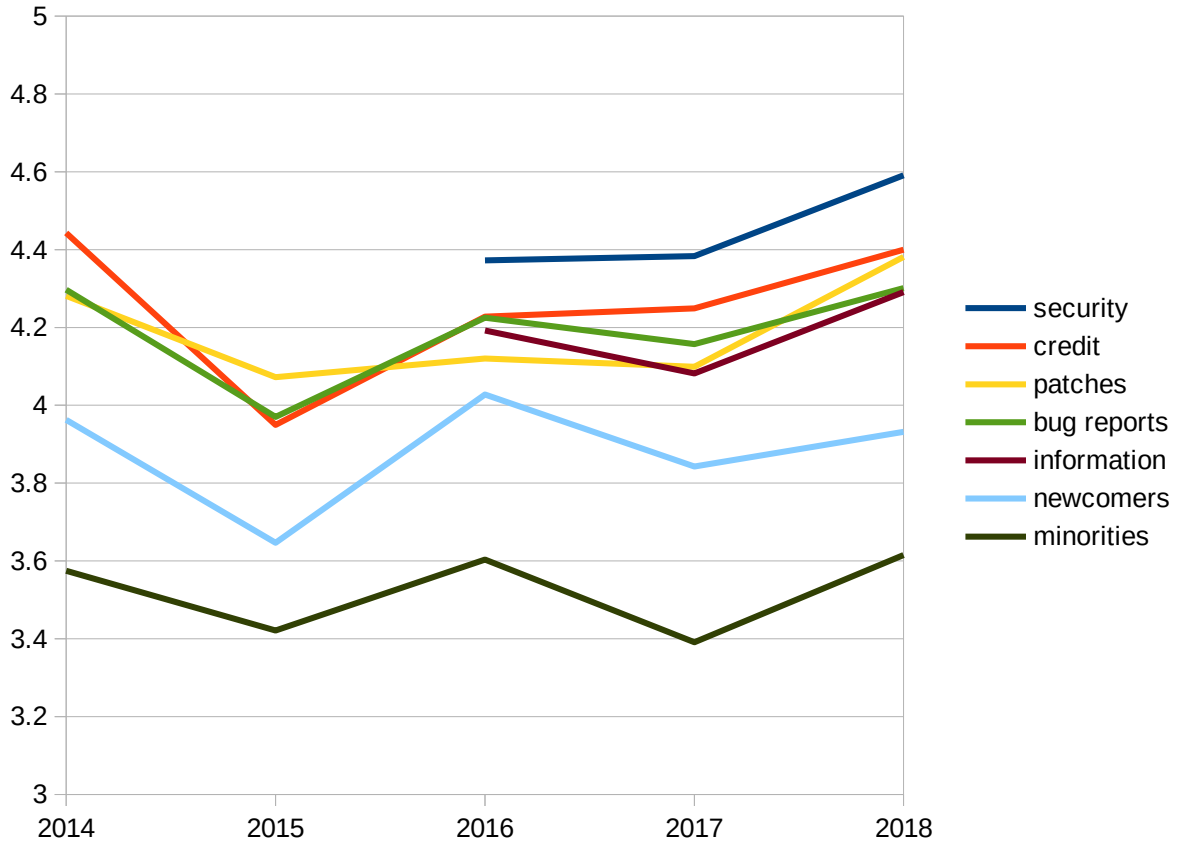
How good is the project to handle…

The respondents grade the areas between 1 (worst) to 5 (best). Here's a chance to check how we have changed in these areas over time. What are the trends?

The individual categories people rated are in security, giving credit, handle patches, deal with bug reports, information, help newcomers and minorities.

This graph below is the average score in each category and how they have changed over the last five years. The order of the categories in the legend shows in top-to-bottom how they range against each other in 2018. We're rated best in handling security issues, we're worst in handling minorities.

The scores are not changing much over the years but small improvements the last two years on several of the questions can be noted.



The average scores 2018 were:

security	4.59
credit	4.4
patches	4.38
bug reports	4.30
information	4.29
newcomers	3.93
minorities	3.6

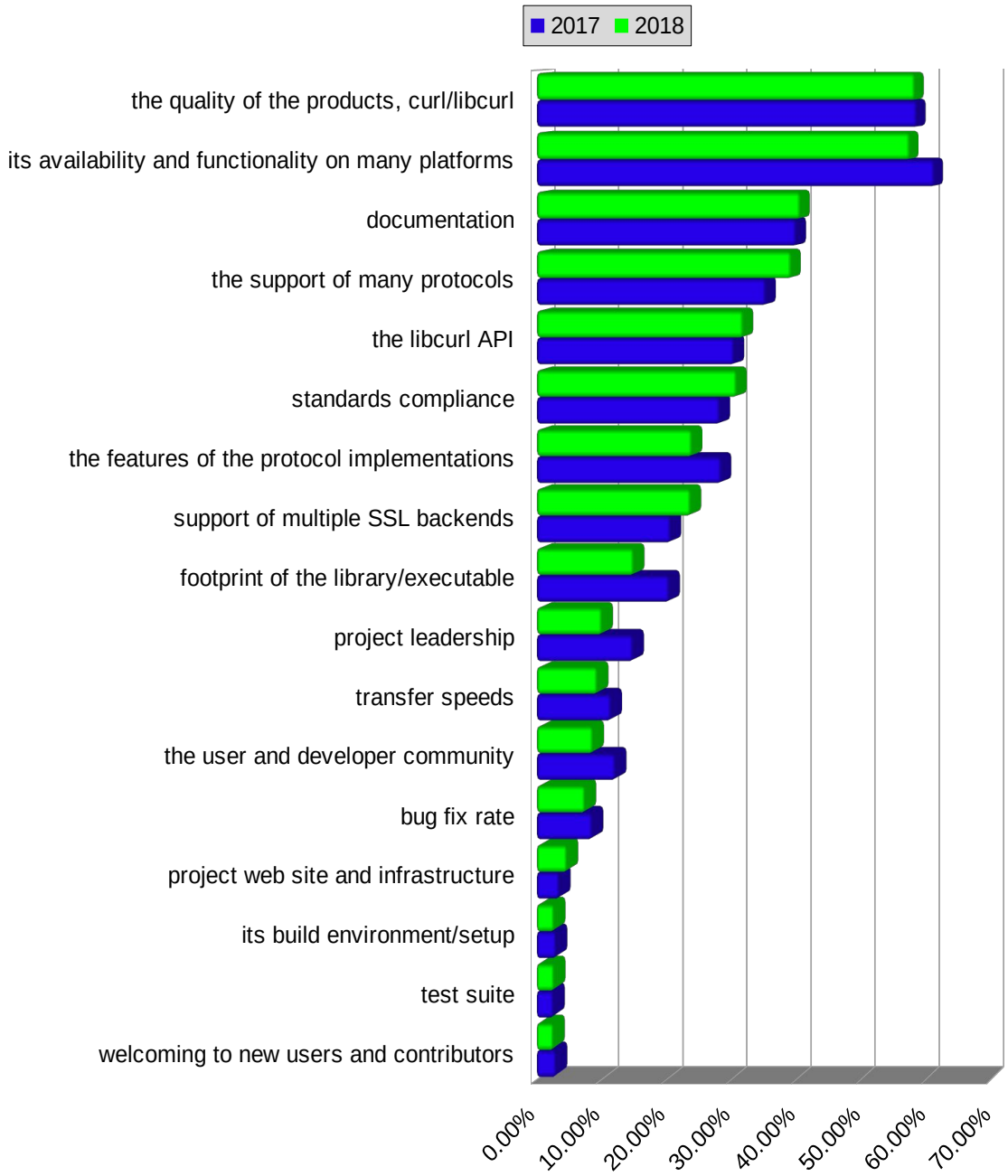
Which are the curl project's best areas?

n = 569

What are the best areas of curl? The primary reasons people use and like curl perhaps? Similar to many other questions, this list doesn't change a lot either. Here are them all shown in a graph compared to last years' results. The two top reasons changed places, and there was some minor changes but the general view remains.

It is quite obvious that our quality and multi-platform abilities are our special sauce, even though multi-platform shrunk from 61.3% to 57.5% this time around. 58.3% checked "quality" as best area.

In the bottom end we can see that very few thinks our test suite or ability to welcome new users and contributors are our best areas.

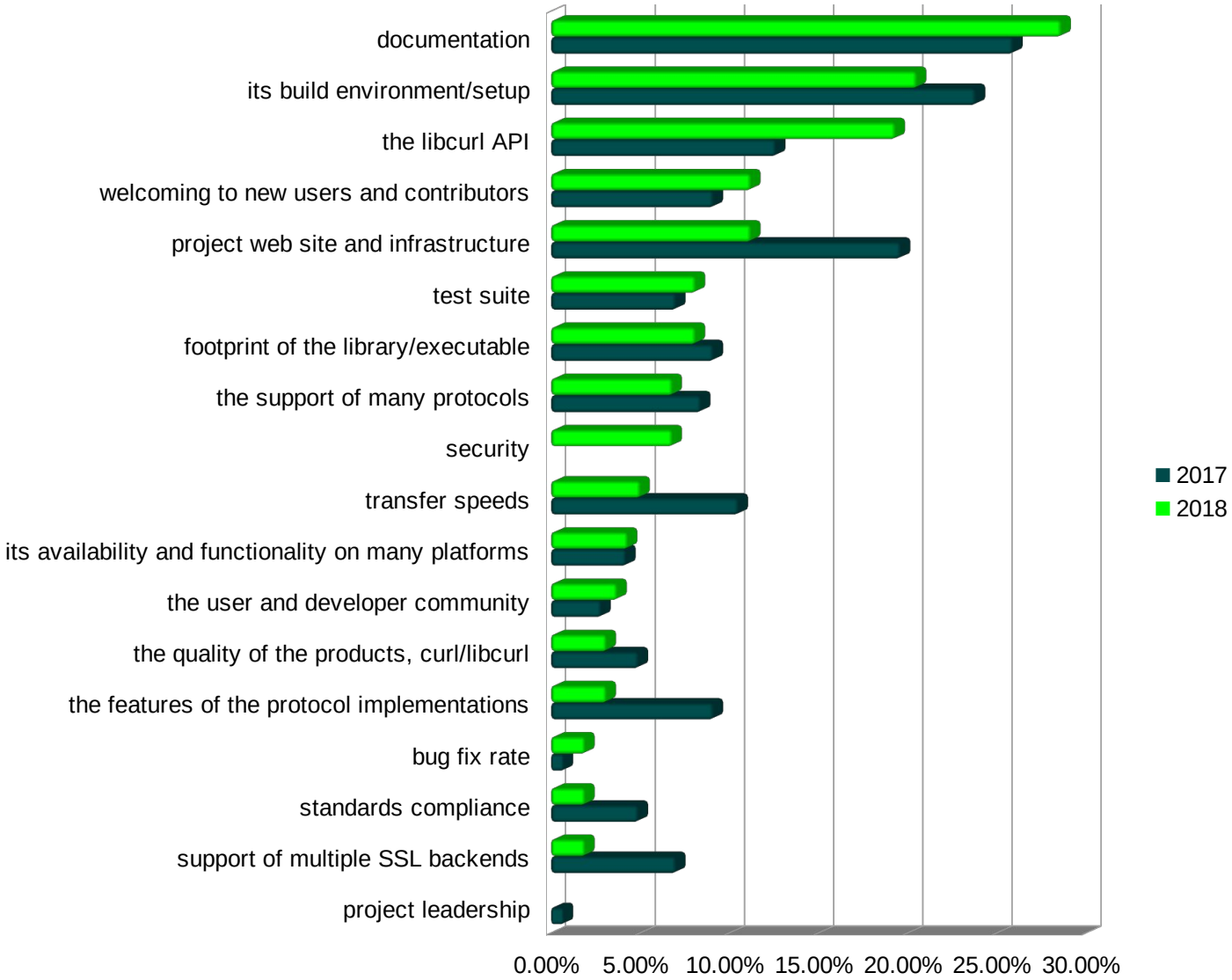


Which are the curl project's worst areas?

n = 161

Much fewer answered to this than the best areas question. This is pretty much the inverse of the previous question. As usual, the top-voted "worst area" is documentation, which also as usual is the 3rd most voted best area, showing the complexity and how hard it is to get documentation right and to satisfy users. The libcurl API is voted the 5th best area and at the same time the 3rd worst..

Perhaps the biggest changes from last year are the most interesting bits in this. Project web site and infrastructure went from 19.6% to 11.20% worst, transfer speeds from 10.5% to 5.0%, features from 9.1% to a mere 3.1%, standards compliance from 4.9% to 1.9% and multiple SSL from 7.0% to 1.9%!



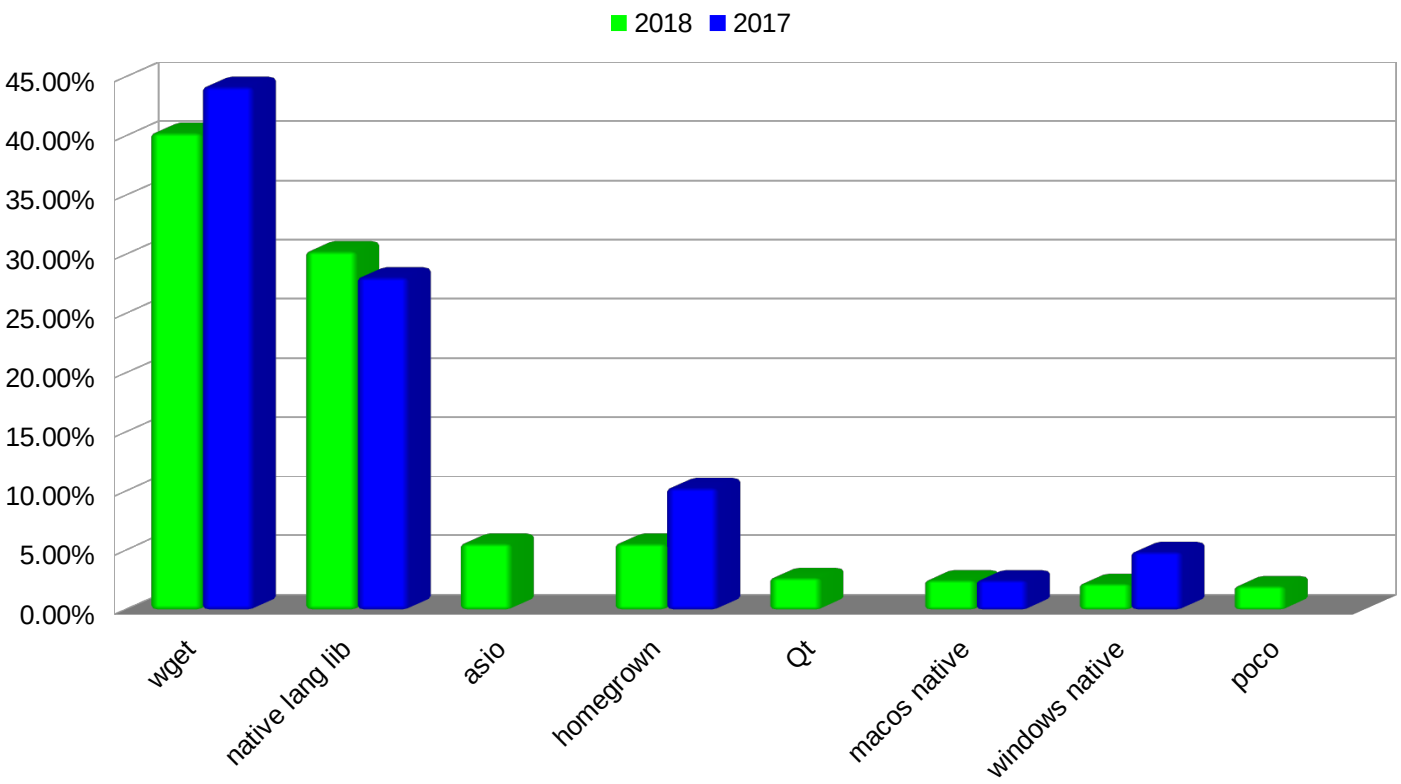
Exactly what goes through people's minds this year as compared to last year is hard to tell here, as certainly we haven't changed much in terms of "multiple SSL support" or "standards compliance" etc. Either the perception changed or this is the result of us having reached a different subset of users than last year…

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

n = 551

This question got a few new alternatives this year, driven by the write-in fields from last year. The results shown in the graph below are all the answers.

Comparing this year to last, with all answers that got 2% of the answers or more:



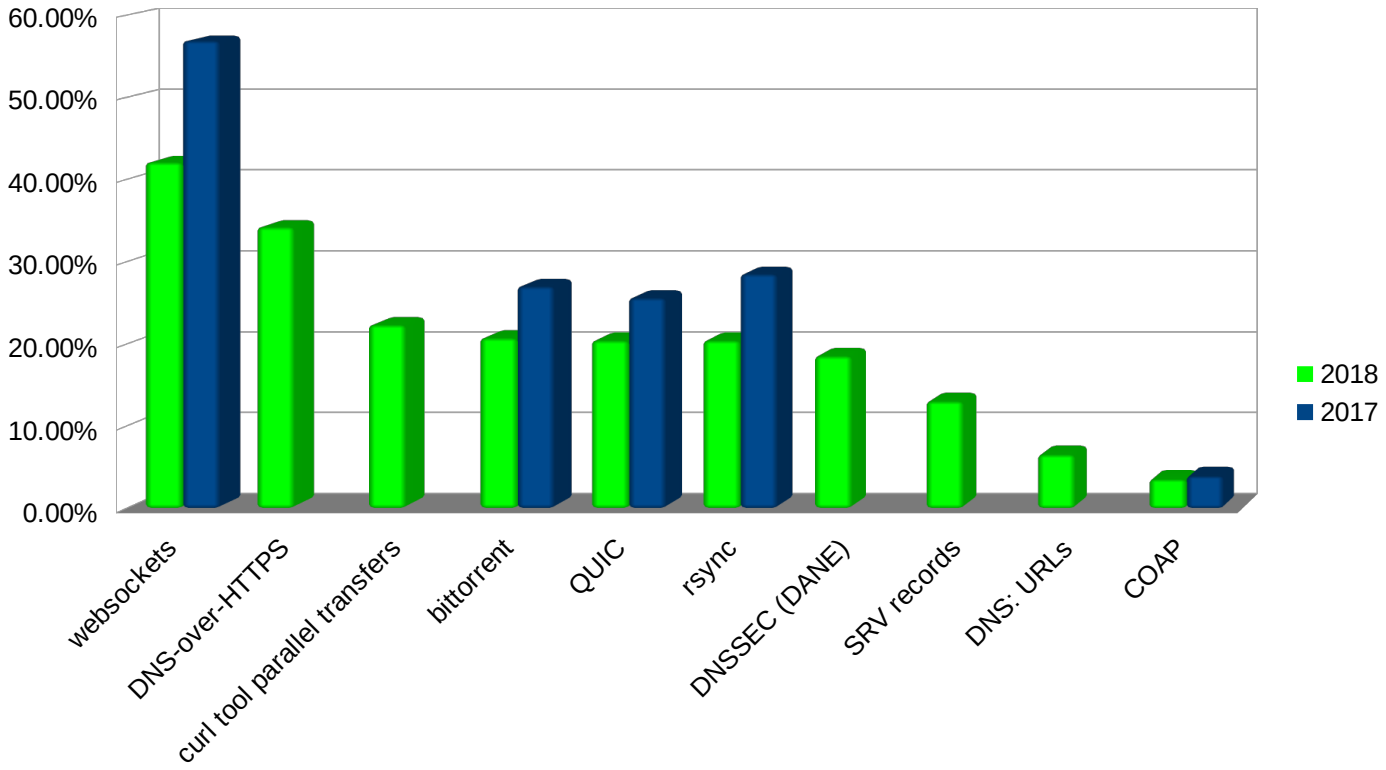
asio, Qt networking and POCO were not options last year so they of course stand alone. Asio obviously being the biggest competitor that isn't "a native language lib", but still it only got 5.6% of the votes. Wget or "code from wget" is by a large margin still what the majority of people think is the alternative (40.3%, while 30.3% selected native language library).

If you miss support for something, tell us what!

n = 368

This question used to ask what protocols that were missing but it was changed to a more generic question this year. I also removed SPDY from the answers in spite of it getting 14.9% of the votes last year. SPDY is just yesterday's protocol… Added alternatives this year include DNS-over-HTTPS, curl tool parallel transfers, DNSSEC and more. It turns out several of the newcomers got quite high interest.

Websockets remain high in demand even if lower than last year, but after that comes two new mentions. DNS-over-HTTPS even already has early code landed in the libcurl-using app doh that I've written as a starting-point to get this done for libcurl: https://github.com/curl/doh



Strikingly, QUIC got *less* votes this year than last even though it is now much closer to reality; down from 27% to 20.7%.

Write-ins on this question included:

- SNMP
- Sieve (RFC 5804)
- IPFS
- REST API
- dat support
- more complete SMB implementation
- Multiple/dynamic SSL backends in *one* libcurl
- I want to be able to get response headers for example like: -w %{h:last-modified}
- UDP proxy for onward QUIC through a forward proxy, wait for spec ;)
- HSTS
- test-suite running in parallel
- SFTP (we already support SFTP!)
- recursive downloads

Closely related to these answers is the next question which asked

If you miss support for something, tell us what!

n = 51

This being an entire free form text field, I had to edit the list somewhat. I've added comments on a bunch of them below.

HTTP Digest Authentication with sha256

Support for RFC 7616 was already added in curl 7.57.0!

Running the test suite on windows

Full ack on this. I would really like to see thins happen. Unfortunately we don't have a lot of active contributors and maintainers on Windows so problems and features that are specific to that platform aren't always getting the attention they need. We will greatly appreciate help and assistance in driving "tests on Windows" forward.

better cmake projects

We just lack people writing the improvements. We know the cmake build setup is lacking in some regards but we don't have enough developers actually working on improving this

Strongly typed variants of curl_easy_setopt

This has been discussed on and off, but hasn't happened yet. One of the reasons perhaps because we haven't been able to figure out the most suitable way this should be done so that all existing functionality can be expressed using a strongly typed version of the API. If you have ideas or suggestions on how you think the API could work in a future, please step forward and help us out!

I would like a library initialization function (similar to curl_global_init) that lets me specify which protocols libcurl should handle and disables all others.

That sounds exactly like `CURLOPT_PROTOCOLS` which has been around since 7.19.4!

Our biggest problem is that we haven't quite figured out how we can distribute a pre-built curl/libcurl binary if we don't know what libraries do/don't exist on customer machines.

This sounds like a wider problem than just a libcurl problem but I understand that this is probably not easy. If you have specific ideas on how we in curl can help make things easier or better, please tell us!

Better API for integrating into an existing event loop.

In my eyes, our existing APIs are pretty good at this, even if the event-based API can be a bit quirky but then I mostly blame that on the nature of events which generally make things harder than otherwise. Here again: if you have specific ideas of what's bad and how it could be made better or easier, please let us know!

Aggregated bandwidth rate limiting for a multi

That would be cool to offer, but is pretty complicated.

Function to clean up idle connections in a multi

This sounds related to the PR Max Dymond works on that can keep connections alive: https://github.com/curl/curl/pull/1641

It would take some similar function like that to discover that an unused connection in the connection pool is actually over its idle limit and close it down. This feature is also already mentioned in the TODO.

Simple header retrieval makes it easy to use in scripts without cut awk grep mess

If we would extend curl's -w option to be able to show the contents of specific headers, that would indeed allow for that in easy ways. I'm cautiously in favor of this idea, especially if implemented purely in curl and not particularly in libcurl.

recursive download

Wow. That's a massive ask that I would be hard to persuade to agree to. The amount of parsing and logic to add for that to become usable is not insignificant while at the same time there exists several good tools already that support this.

Read requests from an har file (http archive) and replay them like a browser in terms of dependencies and parallelism.

Replaying requests from a har file sounds like an interesting idea that I'd very much like to explore further, but I believe the "do it like a browser" is way harder and more complicated than it may sound.

Too easy and commonplace to invisibly disable security (-k, VERIFYPEER/HOST). Would be useful to be able to enforce secure protocols with security features enabled.

We've discussed adding support for an environment variable that if set renders -k useless. This would allow users who want stricter curl invokes to set the environment variable and then no scripts that runs curl could get away with -k uses (see CURL_REFUSE_CLEARTEXT in the TODO). Here's room for you to step up and help us implement this!

Colored terminal output for curl binary in verbose mode to better distinguish between headers, content and connection info.

Lovely. Bold header support has been merged and will be present in curl 7.61.0.

Compilation as C++ code: Add casts for all malloc()/calloc() calls. The reason is that we compile the curl code for .NET.

No. As long as libcurl is C code, we will stick to write C code. There's no good reason for anyone to build libcurl pretending it is C++. C++ and C are not easy interchangeable. That would just add piles of work for us with very little, if any, benefits.

brotli (de)compression support for testing servers (enabled by –compressed)

Done! This has been supported since 7.57.0.

More tooling around client certs - e.g. a default cert if the server accepts the signing CA

This is very vague. “more tooling” that would do what? It is also a bit complicated since curl/libcurl can be built with so many different TLS libraries so any tool we’d build at the same time of curl would then have that or those TLS libraries to work with. Not unsolvable, but that complicates life.

Express your algorithms in the highest-level language possible. Compile that DSL to machine code.

Yikes. That’s not doable.

Client cert handling on the cmdline can be a bit cumbersome...I almost want a list of services in config file to apply the keys to. Think ssh config.

I’m sure client cert handling leaves a lot of room for improvement. Even though lots of users in this survey claim to use it, we get very few bug reports and barely any patches for it. Users of this features should gather and discuss how it can be improved and then we can work on adding features to make sure it gets better.

A curl_multi implementation around poll() instead of select()

Sure, that would even be a fairly easy thing to do. Feel free to submit a pull request! An application can even today avoid select() with the multi interface by using the multi_socket API.

wget's -N flag, because -{z,o}"fname" fname --next… is too verbose

Adding new options that just work as aliases for one or more other options? I’m not convinced that’s a great idea.

Dynamic backend loading like SDL_DYNAMIC_API in SDL2

What backends? And why would they need to be loaded dynamically? Loading files dynamically (by ourselves instead of letting the run-time linker do it) opens up a whole can of worms that we need a very strong reason to take on and I’ve not seen such strong motivations being presented for this case. Please explain!

more environment variables.

For what?

HTTP authentication via callback so URL and headers can be analyzed and send appropriate authentication details

Yes! We’ve had early pieces of code that did this and we’ve discussed it several times in the past but we’ve never seen this concept getting made into reality. I’d love to see someone with a clear use case step forward and help us design this API/setup so that we get it decently right.

A UI like httpie would be awesome

I've not been impressed by httpie's UI the little I've used it. It is much more limited that curl's, plus curl has to consider way more functionality and protocols. But I'm also open for suggestions and patches that improve curl's "UI" and command line handling. Please explain how you envision this would work!

Extended test suite for easy out-of-the-box testing on EBCDIC machines (without need to patch tests)

I'd love that. We're ridiculously short of contributors on such platforms though. What's needed to get this to work? Companies/people on EBCDIC systems need to step forward and ideally also start running automated tests for us to make sure the build remains functional. It is *very* hard to maintain non-ASCII functionality for people on ASCII platforms.

More examples of the APIs

We have 103 stand-alone examples and *every* API man page (more than 400) contains a smaller example snippet. But sure, more examples would be good. Documentation is never good enough, can always improve and we're always ready to receive more examples!

Documentation is generally great but could use examples showing how to configure the library and its handles for very high throughput.

libcurl should not need any particular configuring to perform well. It should always do that.

better rtmp, imap and smtp support. it is difficult to use all of those.

Agreed. They're three protocols with just rudimentary support added. It's a cycle: not so many users of them, so not so many work on improving them, making not so many new users are attracted to start using them...

Multiple/dynamic SSL backends in *one* libcurl

Done. Supported since 7.56.0.

websockets

Seems to be fairly widely requested. Here too I would like someone with a use case to step forward and work with us on how you'd like to see libcurl work for this.

http/2 multiplexing

Since libcurl has supported this for many years already (since 7.44.0) I can only assume this refers to the command line tool which does not support it yet. I want to add parallel transfer support to curl in the future by switching it to use the multi interface internally, and once we do that it should also be fairly easy to enable HTTP/2 multiplexing for the tool.

up to date Android binaries

I would be interested in taking up a wider discussion in the curl community about hosting packages/ binaries for various platforms under the curl web site "umbrella". This to help with the trust issue

and the problem that we now often refer users to third party sites and organizations that we really can't know always host and provide sensible and fine packages. I don't even know if it is possible to host and provide Android binaries in such a way.

automatic proxy configuration on Windows

I would not object to it. Has this slightly problem that on Windows we seem to have many users but fewer who actually contribute with code and help fixing bugs.

Nice to have (for testing other software): custom I/O functions to read/write, i.e. abstraction around a file descriptor

That's just code to add. A little bit of work to make it happen for all the various TLS and SSH backends...

C++ api (classes)

We're not C++ hackers in the curl project. I think leaving binding work for the people who work with the particular language is much better. The fact that no C++ binding has managed to attract many users and become an active and vivid open source project has to me just been a hint that there's not that much demand for such a C++ binding.

Protocol-specific wrappers around curl_easy, with good, secure default options set

libcurl should set secure default options already!

Webdav

This is quite a heavy protocol with to add. I would like to hear a more detailed motivation as to why curl/libcurl needs to do this and how an implementation should be made to work to make it useful to applications. Like with many other things: someone with an actual use case should probably step forward.

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

n = 25

Another free text field and a mere 25 persons could think of something to remove…

protocols that don't really get used

Protocols that don't get used don't get many bug reports == not a particular maintenance burden. This suggestion appeared in several different version, some of them naming specific protocols: RTMP, RTSP, LDAP, SMTP, POP3, IMAP and DICT.

I'm open to discuss how to deprecate specific protocols, but it is important to understand that just because *you* don't use and appreciate the particular protocols, you also typically don't suffer from them much, and for those who actually use them, removing them can become a problem.

The -k flag should go

(in favor of the longer more obvious flag or a flag named for the danger (something like '--disable-certificate-security' to cover all protocols) Too many instructions whack in -k for various (bad) reasons and it should be more of a red flag to those who don't know curl well)

We can just break a bazillion scripts out there, even with good intentions. I think the CURL_REFUSE_CLEARTEXT suggestion as laid out in the TODO is a better approach.

Synchronous API

If you look in the code, the "burden" of providing this is in the neighborhood of just a few hundred lines of code and there are hundreds of not thousands of application using it. Even the curl tool does. Removing this API would not help us much nor will it help any users.

RTMP, but librtmp is dead. if absolutely need to keep, find a way to use ffmpeg's implementation.

This was news to me. If librtmp truly is "dead" then I think it puts this protocol (implementation) as head of the line among the protocols to consider to drop support for.

NSS

We're still counting almost 4% of our users on NSS so I think that would be premature. Also, as long as there are people maintaining the NSS library, the curl code that uses it and it doesn't cause us any trouble, I see very little reason to cut code out.

HTTP/2

Almost half the user population uses HTTP/2. Enough said.

HTTP/0.9 support

I wouldn't be totally against this. I would at the very least be happy to put the support behind an option that disables it by default to users as most users are probably not even aware that curl happily will speak HTTP/0.9.

HTTP Pipelining (too many shaky tests)

I've been wanting to do this for a very long time, but every year in this survey a *shocking* amount of people claim they use it. HTTP/2 and multiplexing is already here and delivers what pipelining never could with even more and better functionality.

CURLOPT_SSL_VERIFYHOST == 1

"When the verify value is 1, curl_easy_setopt will return an error and the option value will not be changed". What else could you possibly want? That it would transparently be used and treated as a 2?

curl.h redefining integer types per platform instead of relying on stdint.h

We could indeed bring up the C standard to support for debate. stdint.h comes with C99 and we still put the part at C89. But I don't think we have many problems with our integer types, do we?

Crappy SSL backends

We've just marked axTLS as unsuitable for use. Any other candidate?

Advertising size reduction options in the build instructions. It makes desktop developers unnecessarily cripple the library because they can.

Deliberately removing docs for things just prevent people from using them is not a method I will approve of. Educating the users in the effects of using said options I think is a much better way. People will always be able to shoot themselves in the foot when given powerful tools. That's a side-effect of offering flexibility.

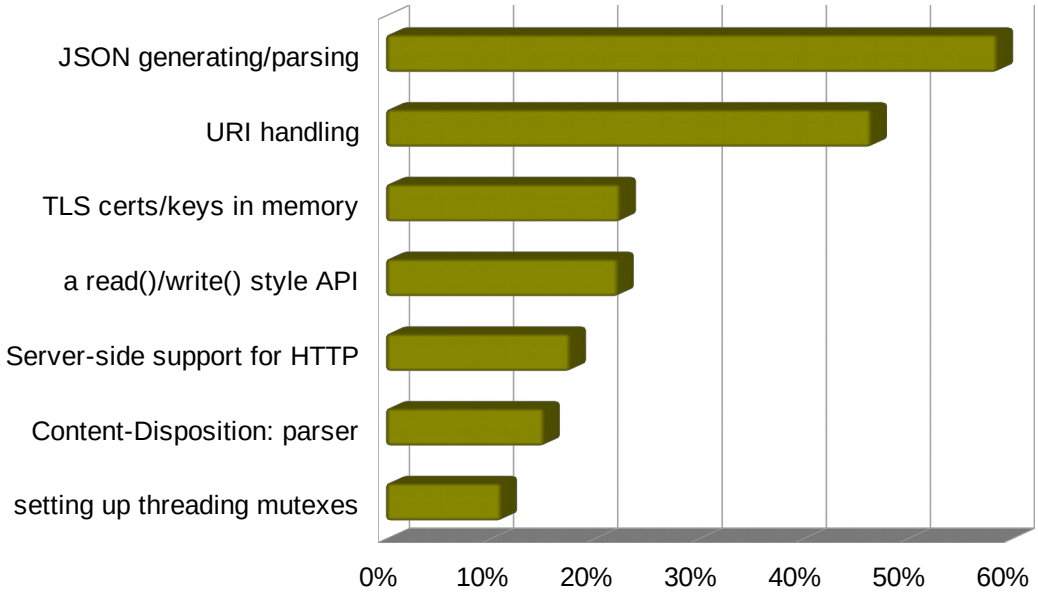
Which of these API(s) would you use if they existed?

n = 395

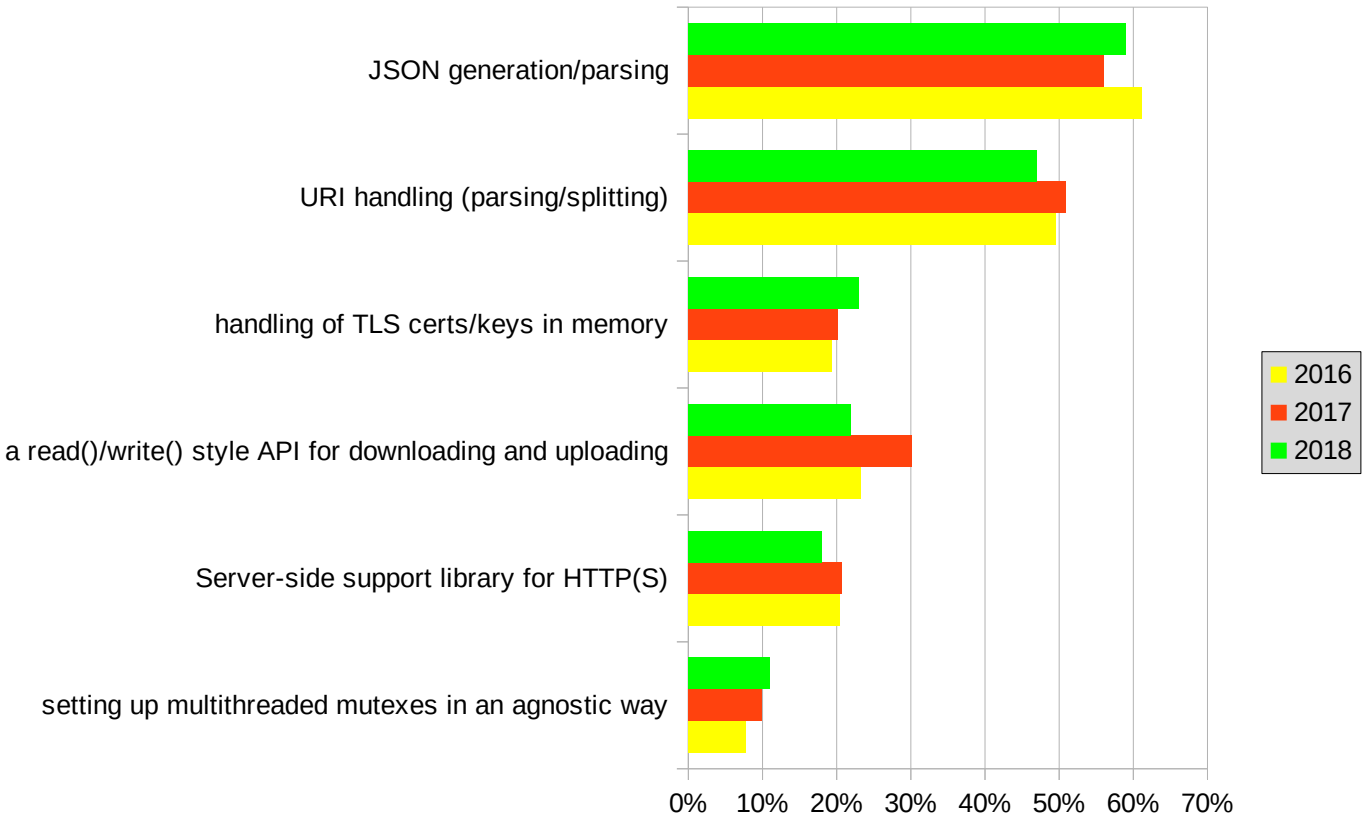
We have features and ideas in our TODO file and we discuss new things every now and then. Here's a question to "feel" the interest in our user population for some of the ideas we have and might implement.

JSON generation and parsing is still very hot apparently, although the discussion about this very subject on the curl-users mailing list a while ago didn't at all show the same level of interest.

I've said it before but I'll say it again: I presented a version of read()/write() API in a separate repository (https://github.com/bagder/fcurl) two years ago but the interest has been basically non-existing. This serves an indication for me that people are eager to be positive in this question, but it might not actually reflect what they want to use or help out to make reality!



… and it is not because the interest changes over time, because it's almost magically stable year over year:



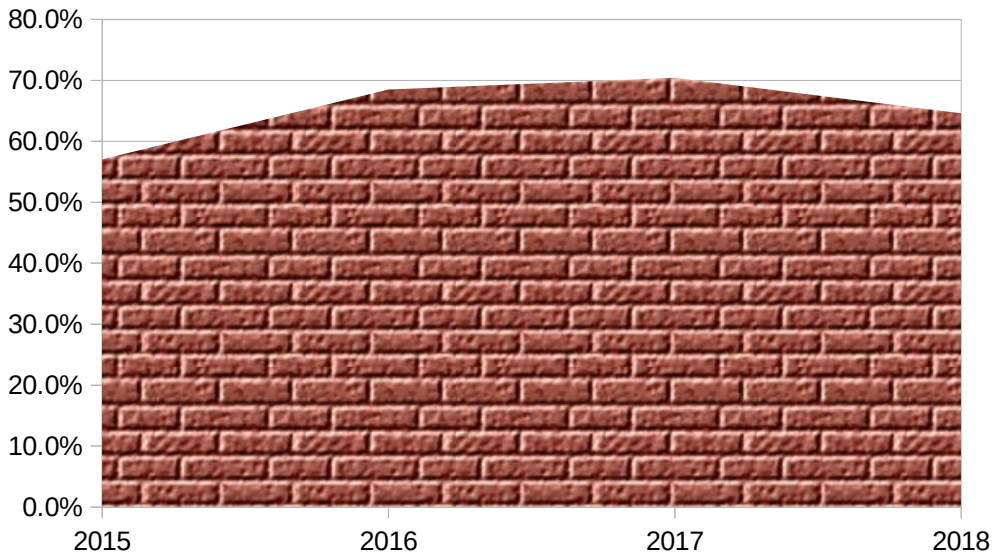
Should curl join an umbrella project?

n = 443

Without any further motivations or explanations, this question has kept getting a firm “no” through all the years we’ve asked. The no share was at 65% this year.

I think however if the discussion opens up and we try to explains some reasons or motivations why we would go either way, I think this question could get a significantly different answer. Right now, I don’t think many users see any particular reason or upside with going into an umbrella organization, but only risk some downsides.

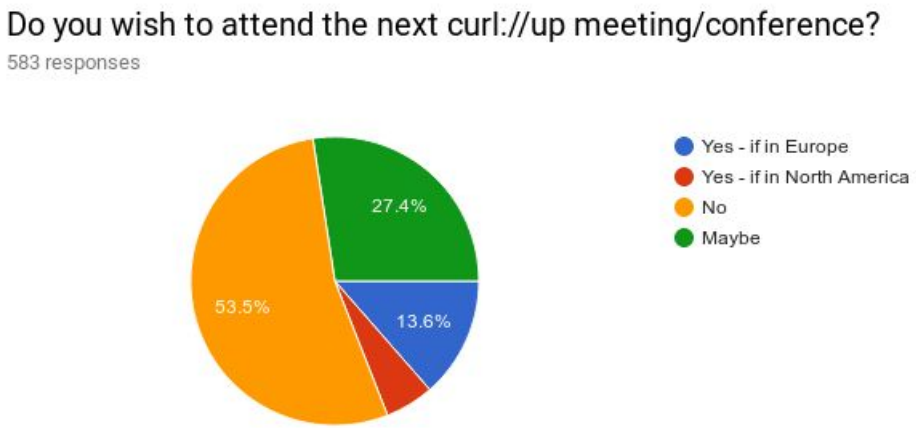
The “no” share in the survey over the last four years:



Do you wish to attend the next curl://up meeting/conference?

n = 583

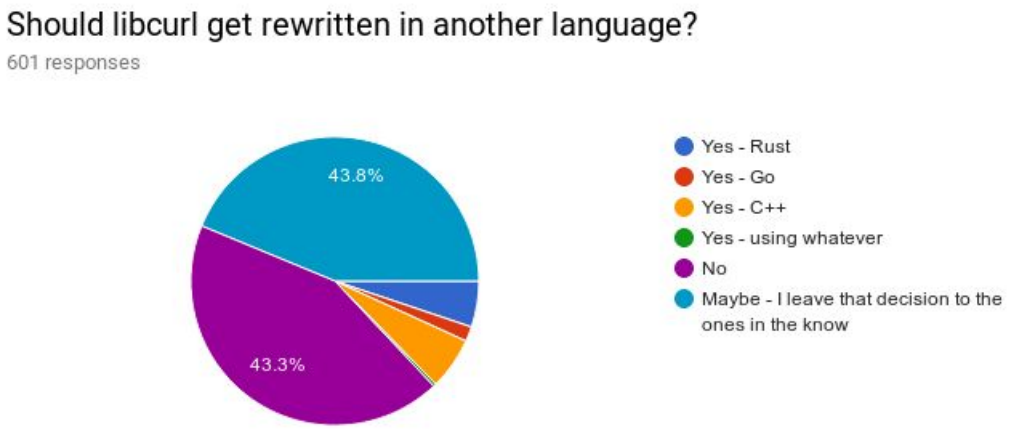
We know we got roughly 3 times as many respondents from Europe as from North America so it's not terribly surprising that we got more than double the rate for Europe (13.6%) than North America (5.5%) in this question! Still, by pure people numbers that's 79 Europeans and 32 North Americans who wishes to attend. I love the enthusiasm!



Should libcurl get rewritten in another language?

n = 601

I inserted this question as a new one this year mostly because this is a popular suggestion to through at me. In particular in times when we release security advisories. There's no effort or real intention to actually rewrite curl into anything, but I figured it would be fun to see how widespread the notion that this is a good idea is.



43.8% abstained and want the ones “in the know” to decide. 43.3% actively said “no”. Among the remaining 12.9% of the users C++ and rust both got basically the same amount: C++ 5,8% and Rust 5.2%. Go only 1.7% while “using whatever” got 0.3%.

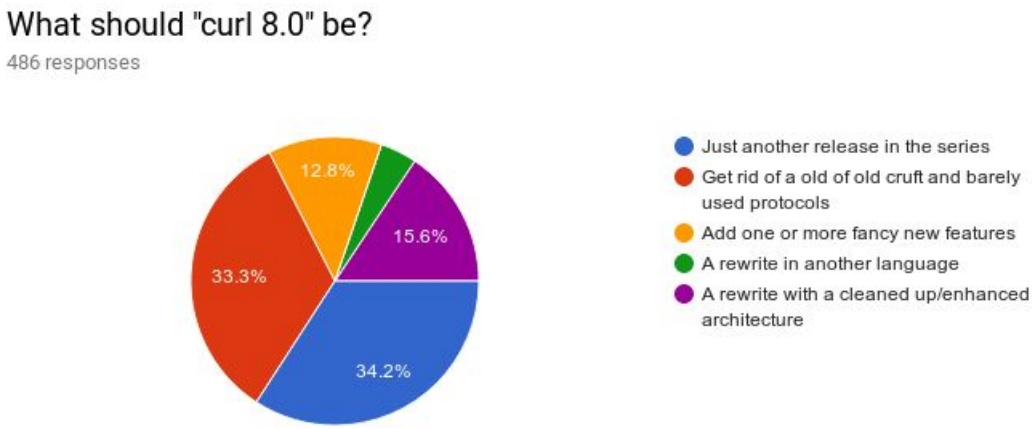
(Personally I can not see the point in rewriting curl in C++ since it is a language with basically the same pitfalls and challenges as C has but with even larger footguns.)

What should "curl 8.0" be?

n = 486

Another one of these bikeshed questions. Easy to have an opinion about.

The largest piece in the pie says “just another release” while the almost as big piece says “get rid of old cruft”. I’m not entirely sure how we will use this information, but I think it at least tells us that there’s no clear opinion on what a version 8 *should be* and that there are at least two widely popular options that possibly can be somewhat combined. That might then be what version 8 will become...



Which question would you like to see in this survey next year?

n = 43

This is a slightly filtered and edited list:

- Would you like to have ONE binary like a go binary is. The reason behind this question is to get rid of the runtime dependencies
- Which protocol do you think needs the most compliance improvements
- Which internal bits of libcurl would you like to see exposed in the API?
- Which architectural improvements would you like to see in the API?
- Where do you think the curl api is unintuitive?
- What is your favorite curl feature?
- What is the most unusual consumer product you have found curl within? My Toshiba TV!
- what are you using curl for (in my case I primarily use it to validate interoperability of haproxy's HTTP/2 implementation and to help reproduce bugs).

- Using space for 8.0 thoughts. Resource intensive, but a total refresh with long support tail 7.x would be sweet.
- to what extent do you understand curl options (i.e. power users, vs just running it in a test suite)? 2. how do you use curl - where's the value to your organization?
- The "How many years have you been using curl?" doesn't have an option for 1 -2 years (which is where I fall).
- Should new features be written in another language?
- Question on desired sources of security-related information used (e.g. OS trust stores, application-provided, embedded SCT, provided SCT, CRLs, OCSP, other sources).
- Probe differences in professional vs personal interactions. In my case, I am on the job, so I can't go much beyond what the company needs, and sometimes very long stretches of time elapse between [lib]curl work - my feet are in other fires. Info about organization worked/volunteered for? (number of employees/developers...) Number of other developers using [lib]curl. Number of projects/products using [lib]curl. The "Best Area" question's limitation of 5 answers was severely limiting. The project has MANY positives!! :) :)
- On a scale of 1-10 where 10 is "you love curl", I'm a 9 btw
- More questions about what kinds of projects curl is actually used for.
- More cowbell?
- Ask for if features such as zero copy (or as little copies as possible, by doing things such as allowing buffers passed to writefunction callbacks to remain allocated until freed by the callback) should be implemented. I would also like to see a way where you could pass curl_multi_socket_action a structure CURLM_SOCKETFUNCTION asks you to store with an event loop's data instead of only having the curl library handle it with it's hash table for a few performance enhancements.
- In which area do you encounter the most problems
- I would like to see less irrelevant questions such as questions regarding how women and minorities are being "handled" because it seems rather condescending to be singled out as a group which needs extra attention in a survey. Why do I need to be "handled" specially if I am a minority. Just handle me like a contributor, tell me when my contributions are crap and judge my contributions based on their merits. Better yet, just ignore any irrelevant attributes altogether. It would be nice to also get a question like a "general comments" section where I could tell you: "Just keep the damn thing simple!"
- How to build for XP would be nice
- how much do you like our new websockets support?
- How many hours per week I save with curl
- How do you use the curl CLI? I use it in tests, debugging REST APIs, etc. How often do you use the curl CLI? Why did you choose to use libcurl instead of other libraries?
- Do you use libcurl, or just the curl command line tool? If so can you answer the questions accordingly. Am just a curl command line tool person. I haven't used libcurl, so unable to answer a lot of the questions
- Do you use curl only for install scripts piping into bash?
- Do we need to keep the 'legacy' protocols? If so, which ones?

- DANIEL FOR PRESIDENT
- Ask about which more uncommon flags are being used. Like -J, or --next
- Ask about curl command UX
- Additional comments: where I would write “cURL license is a good feature”
- a question regarding packaging - brew, deb, conan
- "Is C89 compatibility important to you?" (My answer: yes)
- "How did you find curl?" (Answer: I searched for an open source, sensible C tool to give me HTTPS 'get()' functionality from the command line. And a brief search showed it was the perfect solution. 'CURL' is a great name!