TxtNet Browser

Browse the Web over SMS, no WiFi or Mobile Data required!

TextNet Browser is an Android app that allows anyone around the world to browse the web without a mobile data connection! It uses SMS as a medium of transmitting HTTP requests to a server where a pre-parsed HTML response is compressed using Google's Brotli compression algorithm and encoded using a custom Base-114 encoding format (based on Basest).

In addition, any user can act as a server using their own phone's primary phone number and a Wi-Fi/data connection at the press of a button, allowing for peer-to-peer distributed networks.

Download

See the releases page for an APK download of the TxtNet Browser client. A Google Play release is coming soon.

TxtNet Browser is currently compatible with Android 4.4-13+.

Running Server Instances (uptime not guaranteed)

Country	Phone Number	Notes
United States	+1(913)203-2719	Supports SMS to these countries

Let me know if you are interested in hosting a server instance for your area!

> ⚠️**Please note**: All web traffic should be considered unencrypted, as all requests are made over SMS and received in plaintext by the server!

How it works (client)

This app uses a permission that allows a broadcast reciever to recieve and parse incoming SMS messages without the need for the app to be registered as the user's default messaging app. While granting an app SMS permissions poses a security concern, the code for this app is open source and all code involving the use of internet permissions are compartamentalized to the server module. This ensures that unless the app is setup to be a server, no internet traffic is transmitted. In addition, as the client, SMS messages are only programatically sent to and recieved from a registered server phone number. The app communicates with a "server phone number", which is a phone number controlled by a "server host" that communicates directly over SMS using Android's SMS APIs. Each URL request is sent, encoded in a custom base 114, to the server. Usually, this only requires 1 SMS, but just in case, each message is prepended with an order specifier. When the server receives a request, the server uses an Android WebView component to programatically request the website in a manner that simulates a regular request, to avoid restrictions some services (such as Cloudflare) place on HTTP clients. By doing this, any Javascript can also execute on the website, allowing content to be dynamically loaded into the HTML if needed. Once the page is loaded, only the HTML is transferred back to the recipient device. The HTML is stripped of unnecessary tags and attributes, compressed into raw bytes, and then encoded. Once encoded, the messages are split into 160 character numbered segments (maximizing the GSM-7 standard SMS size) and sent to the client app for parsing and displaying.

Side note: Compression savings have been estimated to be an average of 20% using Brotli, but oftentimes it can save much more! For example, the website `example.com` in stripped HTML is 285 characters, but only requires 2 SMS messages (189 characters) to receive. Even including the 225% overhead in data transmission, it is still more efficient!

Why encode the HTML in the first place?

SMS was created in 1984, and was created to utilize the extra bytes from the data channels in phone signalling. It was originally conceived to only support 128 characters in a 7-bit alphabet. When further characters were required to support a subset of the UTF-8 character set, a new standard called UCS-2 was created. Still limited by the 160 bytes available, UCS-2 supports more characters (many of which show up in HTML documents) but limits SMS sizes to 70 characters per SMS. By encoding all data in GSM-7, more data can be sent per SMS message than sending the raw HTML over SMS. It is possible that it may be even more efficient to create an encoding system using all the characters available in UCS-2, but this limits compatibility and is out of the scope of the project.

Server Hosting

TxtNet Browser has been rewritten to include a built-in server hosting option inside the app. Instead of the now-deprecated Python server using a paid SMS API, any user can now act as a server host, allowing for distributed communication.
To enable the background service, tap on the overflow menu and select "TxtNet Server Hosting". Once the necessary permissions are granted, you can press on the "Start Service" toggle to initialize a background service.
TxtNet Server uses your primary mobile number associated with the active carrier subscription SIM as a number that others can add and connect to.
Please note that this feature is still in early stages of development and likely has many issues. Please submit issue reports for any problems you encounter.
For Android 4.4-6.0, you will need to run adb commands one time as specified in the app. For Android 6.0-10.0, you may also use Skizuku, but a PC will still be required once. For Android 11+, no PC is required to activate the server using Shizuku.

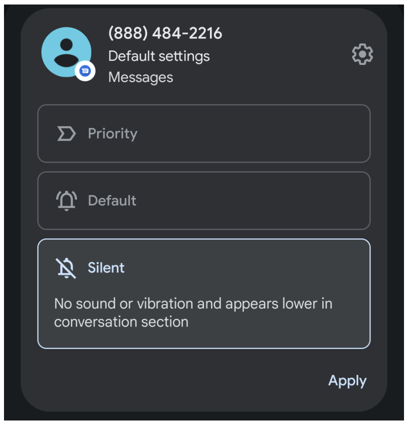
Desktop Server Installation (Deprecated)

~~The current source code is pointed at my own server, using a Twilio API with credits I have purchased. If you would like to run your own server, follow the instructions below: 1. Register for an account at [Twilio](https://twilio.com/), purchase a toll-free number with SMS capability, and purchase credits. (This project will not work with Twilio free accounts) 2. Create a Twilio application for the number. 3. Sign up for an [ngrok](http://ngrok.com/) account and download the ngrok application 4. Open the ngrok directory and run this command: `./ngrok tcp 5000` 5. Visit the [active numbers] (https://console.twilio.com/US1/develop/phone-numbers/manage/incoming) page and add the ngrok url to the "A Message Comes In" section after selecting "webhook". For example: "https://xyz.ngrok.io/receive_sms" 6. Download the TxtNet Browser [server script] (https://github.com/lukeaschenbrenner/TxtNet-Browser/blob/master/SMS_Server_Twilio.py) and install all the required modules using "pip install x" 7. Add your Twilio API ID and Key into your environment variables, and run the script! `python3 ./SMS_Server_Twilio.py` 8. In the TxtNet Browser app, press the three dots and press "Change Server Phone Number". Enter in the phone number you purchased from Twilio and press OK!~~

FAQ/Troubleshooting

Bugs:

- Many carriers are unnecessarily rate limiting incoming text messages, so a page may look as though it "stalled" while loading on large pages. As of now the only way to fix this is to wait!
- In congested networks, it's possible for a mobile carrier to drop one or more SMS messages before they are recieved by the client. Currently, the app has no logic to mitigate this issue, so any websites that have stalled for a significant amount of time should be requested again.
- In Android 12 (or possibly a new version of Google Messages?), there is a new and "improved" messages blocking feature. This results in no SMS messages getting through when a number is blocked, which makes the blocking feature of TxtNet Browser break the app! Instead of blocking messages, to get around this "feature", you can silent message notifications from the server phone number.

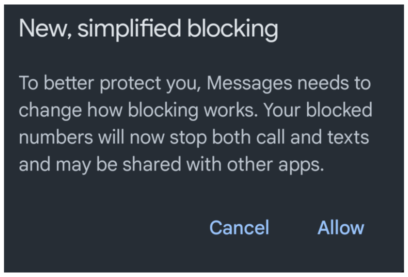

(888) 484-2216
Default settings
Messages

Priority

Default

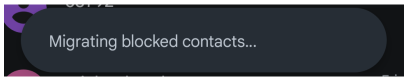
Silent
No sound or vibration and appears lower in conversation section

Apply

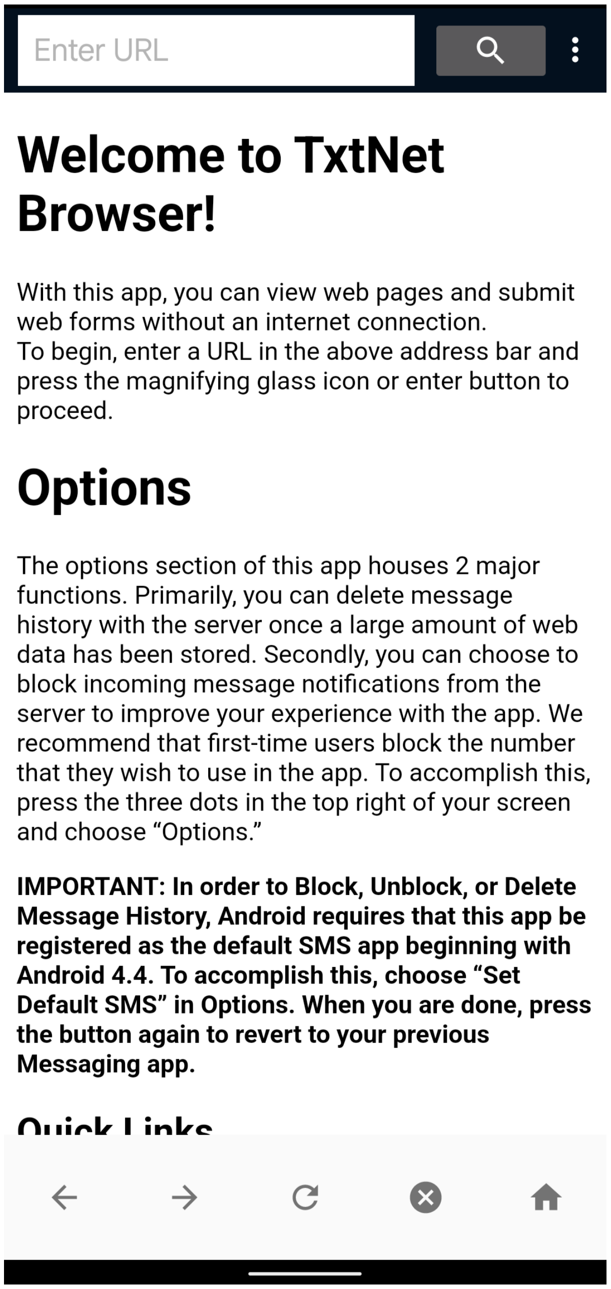

New, simplified blocking

To better protect you, Messages needs to change how blocking works. Your blocked numbers will now stop both call and texts and may be shared with other apps.

Cancel Allow


Migrating blocked contacts...

Screenshots (TxtNet 1.0)


Enter URL

Welcome to TxtNet Browser!

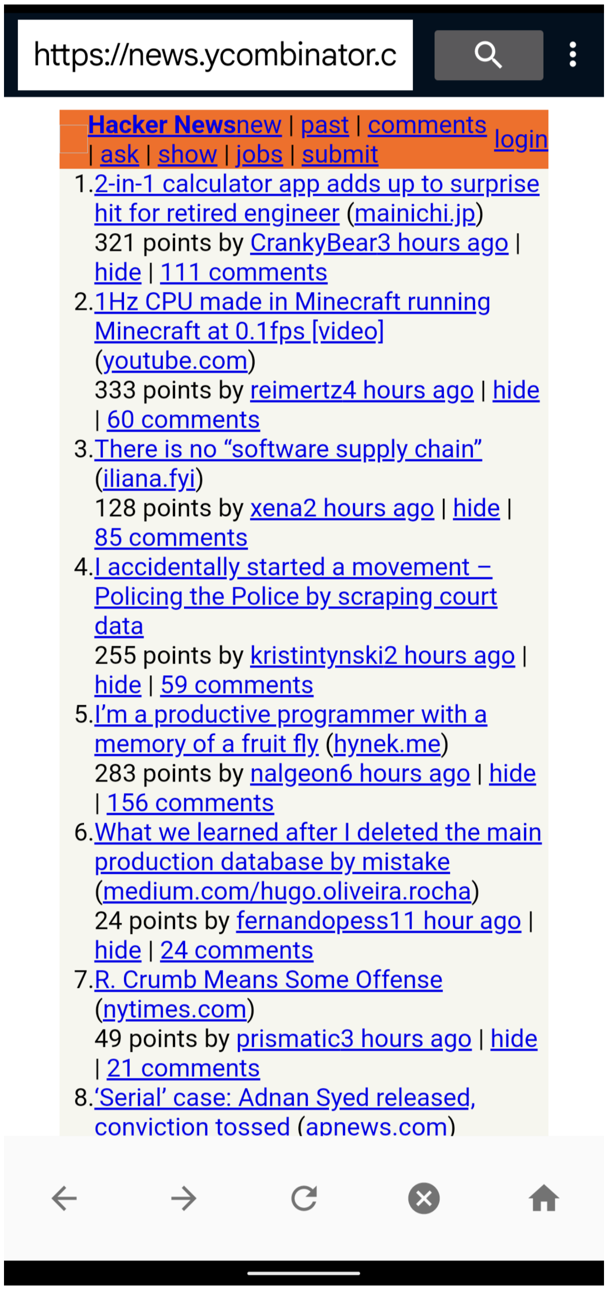
With this app, you can view web pages and submit web forms without an internet connection.
To begin, enter a URL in the above address bar and press the magnifying glass icon or enter button to proceed.

Options

The options section of this app houses 2 major functions. Primarily, you can delete message history with the server once a large amount of web data has been stored. Secondly, you can choose to block incoming message notifications from the server to improve your experience with the app. We recommend that first-time users block the number that they wish to use in the app. To accomplish this, press the three dots in the top right of your screen and choose "Options."

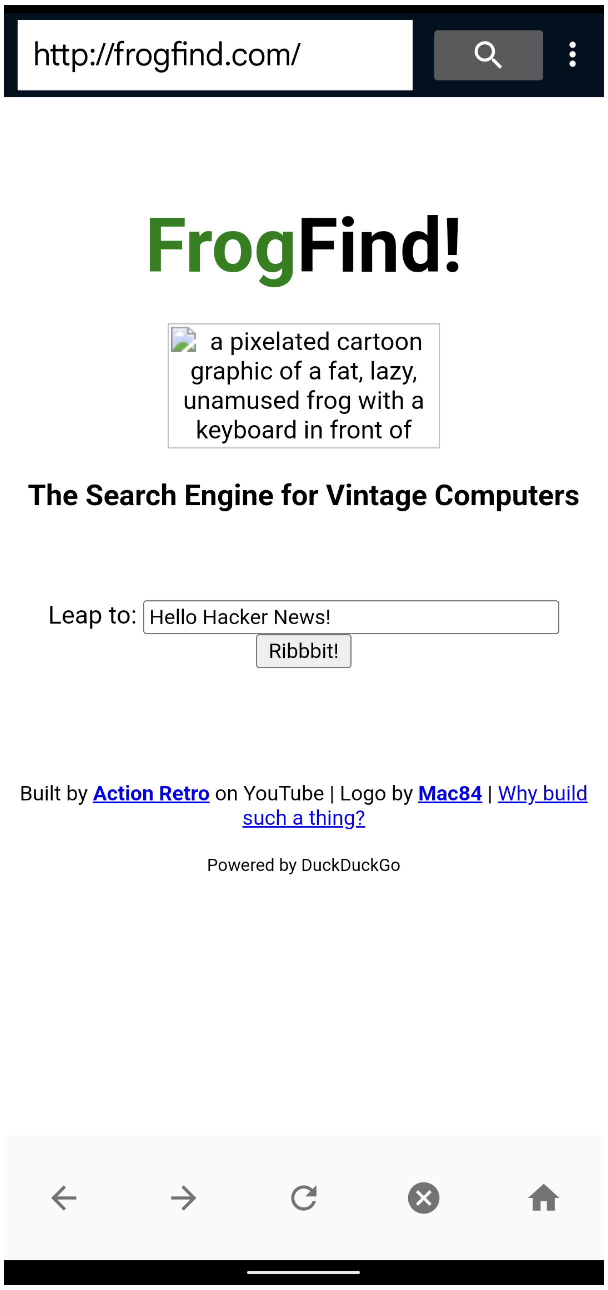
IMPORTANT: In order to Block, Unblock, or Delete Message History, Android requires that this app be registered as the default SMS app beginning with Android 4.4. To accomplish this, choose "Set Default SMS" in Options. When you are done, press the button again to revert to your previous Messaging app.

Quick Links


https://news.ycombinator.c

Hacker Newsnew | past | comments | ask | show | jobs | submit login

1. 2-in-1 calculator app adds up to surprise hit for retired engineer (mainichi.jp)
321 points by CrankyBear3 hours ago | hide | 111 comments
2. 1Hz CPU made in Minecraft running Minecraft at 0.1fps [video] (youtube.com)
333 points by reimertz4 hours ago | hide | 60 comments
3. There is no "software supply chain" (iliana.fyi)
128 points by xena2 hours ago | hide | 85 comments
4. I accidentally started a movement – Policing the Police by scraping court data
255 points by kristintynski2 hours ago | hide | 59 comments
5. I'm a productive programmer with a memory of a fruit fly (hynek.me)
283 points by nalgeon6 hours ago | hide | 156 comments
6. What we learned after I deleted the main production database by mistake (medium.com/hugo.oliveira.rocha)
24 points by fernandopess11 hour ago | hide | 24 comments
7. R. Crumb Means Some Offense (nytimes.com)
49 points by prismatic3 hours ago | hide | 21 comments
8. 'Serial' case: Adnan Syed released, conviction tossed (apnews.com)


http://frogfind.com/

FrogFind!

a pixelated cartoon graphic of a fat, lazy, unamused frog with a keyboard in front of

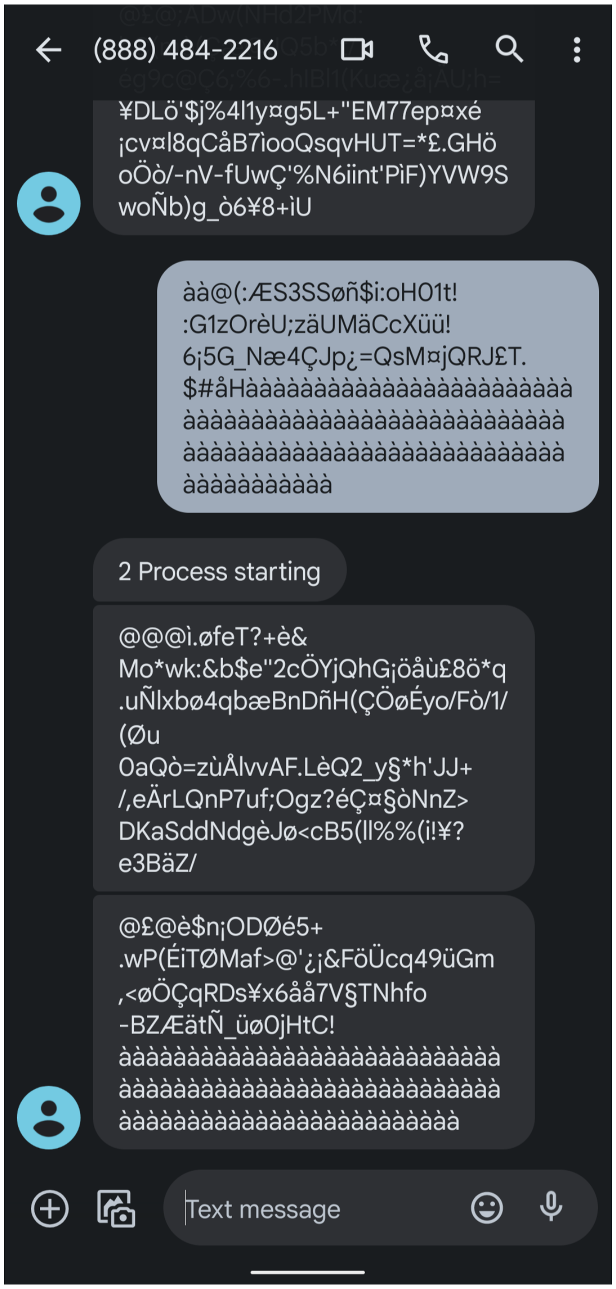
The Search Engine for Vintage Computers

Leap to: Hello Hacker News!

Ribbbit!

Built by **Action Retro** on YouTube | Logo by **Mac84** | Why build such a thing?

Powered by DuckDuckGo


(888) 484-2216

```
@£@;ADw(NHd2PMd:
ég9c@Ç6;%6-.hIBl1(Kuæ¿å¡AU;h=
¥DLö'$j%4l1y¤g5L+"EM77ep¤xé
¡cv¤l8qCåB7ìooQsqvHUT=*£.GHö
oÖò/-nV-fUwÇ'%N6iint'PìF)YVW9S
woÑb)g_ò6¥8+ìU
```

```
àà@(:ÆS3SSøñ$i:oH01t!
:G1zOrèU;zäUMäCcXüü!
6¡5G_Næ4ÇJp¿=QsM¤jQRJ£T.
$#åHààààààààààààààààààààààààààà
àààààààààààààààààààààààààààààà
àààààààààààààààààààààààààààààà
àààààààààààà
```

2 Process starting

```
@@@ì.øfeT?+è&
Mo*wk:&b$e"2cÖYjQhG¡öåù£8ö*q
.uÑlxbø4qbæBnDñH(ÇÖøÉyo/Fò/1/
(Øu
0aQò=zùÅlvvAF.LèQ2_y§*h'JJ+
/,eÄrLQnP7uf;Ogz?éÇ¤§òNnZ>
DKaSddNdgèJø<cB5(ll%%(i!¥?
e3BäZ/
```

```
@£@è$n¡ODØé5+
.wP(ÉiTØMaf>@'¿¡&FöÜcq49üGm
,<øÖÇqRDs¥x6åå7V§TNhfo
-BZÆätÑ_üø0jHtC!
àààààààààààààààààààààààààààà
àààààààààààààààààààààààààààà
ààààààààààààààààààààààààà
```

Text message

Demo (TxtNet 1.0)

demo.mp4

0:00

> Demo video shown above

Development

🚧 If you are skilled in Android UI design, your help would be greatly appreciated! 🚧 A consistent theme and dark mode would be great additions to this app.

Feel free to submit pull requests! I am a second-year CS student with basic knowledge of Android Development and Server Development, and greatly appreciate help and support from the community.

Future Impact

My long-term goal with this project is to eventually reach communities where such a service would be practically useful, which may include:

- Those in countries with a low median income and prohibitively expensive data plans
- Those who live under oppressive governments, with near impenetrable internet censorship

If you think you might be able to help funding a local country code phone number or server, or have any other ideas, please get in contact with the email in my profile description!

License

GPLv3 - See LICENSE.md

Credits

Thank you to everyone who has contributed to the libraries used by this app, especially Brotli and Basest. Special thanks goes to Coldsauce, whose original project Cosmos Browser was the original inspiration for this project!
My original reply to his Hacker News comment is here. In addition, I would like to thank Zachary Wander from XDA for their excellent Shizuku implementation tutorial and Aayush Atharva for the amazing foundation they created with Brotli4J, allowing for a streamlined forking process to create the library BrotliDroid used in this app.