We are living in instant messenger hell

I had to install WhatsApp, because some friends are refusing to communicate in any other way, which made me realise how tired and disillusioned I am when I have to face yet another instant messenger network - at least, with some work, Pidgin can still connect to more or less everything and anything.

Note: I have updated some parts of this entry. This is due to the fact that I wrote about XMP without spending enough time exploring what it's really capable of, for which I'm sorry. I made changes to my article according to these finds.

Me vs. IM

Before the dawn of the always online era (pre 2007) the world of instant messengers was completely different. For me, it all started with various IRC[^1] rooms, using mIRC[^2], later extended with ICQ[^3] in 1998.

I loved ICQ. I loved it's notifications sound - *I have it as notification sound on my smartphone and it usually results in very confused expressions from people who haven't heard the little 'ah-oooh' for a decade* -, it's capability of sending and receiving files, the way you could search for people based on location, interest tags, etc.

> The sixth protocol version appeared in ICQ 2000b and faced a complete rework. Encryption was significantly improved. Thanks to the new protocol, ICQ learned how to call phones, and send SMS and pager messages. Users also got the option of sending contact requests to other users.[^4]

Around this time, Windows included an instant messenger in their operating systems: MSN Messenger[^5], later renamed to Windows Live Messenger. It was inferior, but because it was built in to Windows, it took all the ICQ users away. It's completely dead now.

The multiplication of messengers had one useful effect though: people who got fed up running multiple clients for the same purpose - to message people - came up with the idea if multi-protocol applications. I used Trillian[^6] for many years, followed by Pidgin[^7] once I switched to linux.

With the help of these multi-protocol miracles it wasn't an issues when newcomers like Facebook or Google released their messaging functionality: both were built in top of XMPP[^8], an open standard for instant messaging, and they were both supported out of the box in those programs.

Around this time came Skype and it solved all the video call problems with ease. It was fast, p2p, encrypted, ran on every platform, supported more or less everything people needed, including multiple instances for multiple accounts. Skype was on a good way to eliminate everything else. Unfortunately none of the multi-protocol messengers ever had a native support to it: it only worked if a local instance of Skype was running.

A few years later iPhone appeared and it ate the consumer world; not long before that, BlackBerry did the same to the business. Smartphones came with their own, new issues: synchronization, and resource (battery and bandwidth) limitations. *ICQ existed for Symbian S60, Windows CE, and a bunch of other, ancient platforms, but by the time iPhones and BlackBerries roamed the mobile land, it was in a neglected state in AOL and missed a marvellous opportunity.*

Both of those problems were known and addressed in the XMPP specification. The protocol was low on resources by design, it supported device priority, and XEP-0280: Message Carbons[^9] took care of delivering messages to multiple clients. There was a catch though: none of the well known XMPP providers supported any of these additions, so you ended up using either your mobile device or your computer exclusively at the same time. Most of the big system - AOL, Yahoo!, MSN, Skype, etc - didn't even have a client for iOS, let alone for Android that time.

This lead to a new type of messenger generation: mobile only apps. WhatsApp[^10], BlackBerry Messenger[^11], Viber, etc - none of them offered any real way to be used from the desktop, and they all required

- they still do - a working mobile phone number even to register.

For reasons I'm yet to comprehend, both Google and Facebook abandoned XMPP instead of ~~extending~~ fully implementing it. Google went completely proprietary and replaced gtalk[^12] with Hangouts[^13]; Facebook started using MQTT[^14] for their messenger applications. Both of them were simple enough to be reverse engineered and added to libpurple, but they both tried to reinvent something that already existed.

For Skype, this was a turning point: it was bought by Microsoft, and they slowly moved it from p2p to a completely centralised webapp. The official reasoning included something about power hungry p2p connections... Soon, Skype lost all of it's appeal from it's previous iterations: video and voice was lagging, it was consuming silly amount of resources, it was impossible to stop it on Android, etc. Today, it resembles nothing from the original, incredible, p2p, secure, decentralised, resource-aware application it used to be.

I had to install WhatsApp yesterday - I resisted it as long as I could. It completely mangled competition in the UK and the Netherlands: nobody is willing to use anything else, not even regular text (SMS) or email. It did all this despite it's lack of multi-device support, and the fact that it's now owned by one of the nastiest, people-ignorant businesses around the globe[^15].

So, all together, in February 2018, for work and personal communication, I need to be able to connect to:

- IRC
- Skype

- Facebook
- Telegram
- XMPP
- Workplace by Facebook[^16]
- WhatsApp
- ICQ*
- Google Hangouts*
- WeChat**

* *I still have some contacts on ICQ, though it's a wasteland, and I can't even remember the last time I actually talked to anyone on it. This sort of applies to Hangouts: those who used to use it are now mostly on Facebook*.

** *WeChat is, so far, only a thing if you have Chinese contacts or if you live in/visit China. It's dominating China so far that other networks, like QQ, can be more or less ignored, but WeChat is essential.*

If I install all those things on my phone, I'll run out of power in a matter of hours and the Nomu has an internal 5000mAh brick. They will consume any RAM I throw at them, and I don't even want to think about the privacy implications: out of curiosity I checked the ICQ app, but the policy pushed into my face on the first launch is rather scary. As for Facebook: I refuse to run Facebook in any form on my phone apart from 'mbasic', the no javascript web interface.

Typing on a touchscreen inefficient, and I'm very far from being a keyboard nerd; my logs will be application specific and probably not in any readable/parsable format.

On top of all this, a few days ago Google announced Google Hangouts Chat[^17]. Right now, Google has the following applications to cover text, voice, and video chat:

- Hangouts
- Allo
- Duo
- Hangouts Meet
- Hangouts Chat

That's 5 applications. 5. Only from Google.

Words for the future

I really, really want one, single thing, which allows me:

- native voice and video
- private and group messaging
- multiple concurrent login from varios platforms
- libpurple plugin option
- all devices get all messages

I sort of liked is Telegram[^18]: cross device support, surprisingly fast and low on resources, but it gets attacked because they dared to roll their own crypto, and, in the end, it's still a centralised service, ending up as just another account to connect to, and just another app to run. Since I wrote this entry, a few has tried to point out, that Telegram is not better, than WhatsApp or Signal, but I have to disagree. Yes, WhatsApp is encrypted by default - this also means I need to run my phone as a gateway all the time. No phone = no desktop user. The desktop "app" is a power and resource eater Electron app.

Others asked about Signal. It's doing encryption the paranoid, aggressive way, but the same time, it depends on Google Services Framework on Android, Twilio, AWS, requires a smartphone app, eliminates 3rd party client options, and will only run the "desktop" Electron app if you pair it with a phone app - in which case it's very similar to WhatsApp. Like it or not, it's also a silo, with centralised services, even though you could, in theory, be able to install a complicated server of your own, that relies on the services listed above. It might be better, then WhatsApp, definitely not better from a usability point of view, than Telegram. Privacy wise... unless I can run my own server, without those kind of dependencies, no, thanks - it's just another silo.

I also believe OTR-like encryption is overrated, or at least not as important as many presses. Most of the messages will tell you less, than their metadata, so what's the point? Most of the encryption protocols are exclusive per connected client, meaning you can't have multiple devices with the same account exchanging the same messages - hence the need for the phone apps as gateways. XMPP with OMEMO[^19] is tacking this - if that's on by default, that could work. *Note: TLS, infrastructure level encryption is a must, that is without question.*

While Matrix[^20] looks promising, it's an everything-on-HTTP solution, which I still find odd and sad. HTTP is not a simple protocol - yes, it's omnipresent, but that doesn't make it the best for a particular purpose. There's another problem with it: no big player bought in which could bring the critical mass of users, and without that, it's practically impossible to get people to use it.

Video and voices calls are, in general, in a horrible shape: nearly everything is doing WebRTC, which, while usually works, is a terrible performer, insanely heavy on CPU, and, most of the time, always tries to go for the highest quality, consuming bandwidth like there is no tomorrow.

All this leaves me with **XMPP** and **SIP**.

XMPP is and could be able to cover everything, and, on top of it, it's federated, like email: anyone can run their own instance. I'm still a fan of email (*yes, you read that right*), and a signficant part of it is due to the options you can choose from: providers, clients, even being your own email service.

Unlike with most solutions and silos, the encryption problem (*namely that if encryption is on, only one of the devices can get the messages, our you need to use a router device, like WhatsApp does*) is covered and done with the XMPP extension OMEMO[^21]. It's a multi-client encryption protocoll, that allows simultaneous devices to connect and encrypt at once.

In case of XMPP, voice and video could be handled by a P2P protocol, Jingle[^22], but, unfortunately, it's rarely supported. On Android, I found Astrachat[^23] which can do it, but it lacks many features when it comes to text based communications, unlike Conversations[^24]. On desktop, I'm having serious problems getting Pidgin use video, so not everything is working yet.

This is where SIP comes in: an old, battle tested, proven VOIP protocol, which, so far, worked for me without any glitch in 2018. A few years ago many mobile providers were blocking SIP (among other VOIP protocols), but it's getting much better. Unfortunately I have not started running my own VOIP exchange yet, and ended up using Linphone[^25] as software and provider - for now. The unfortunate part of SIP is that Pidgin doesn't support it in any form.

There is one, very significant problem left: conformist people. I understand WhatsApp is simple and convenient, but it's a Facebook owned, phone only system.

I'd welcome thoughts and recommendations on how to make your friends use something that's not owned by Facebook.

Until then, I'll keep using Pidgin, with a swarm of plugins that need constant updating.

Adding networks to Pidgin (technical details)

Pidgin, which I mentioned before, is a multi protocol client. Out of the box, it's in a pretty bad shape: AIM, MSN, and Google Talk are dead as doornail, most of the systems it supports are arcane (eg. Zephyr) or sort of forgotten (ICQ). The version 3 of pidgin, and it's library, libpurple, has been in the making for a decade and it's still far ahead; the current 2.x line is barely supported.

There is hope however: people keep adding support for new systems, even to ones without proper or documented API.

For those who want to stick to strictly text interfaces, Bitlbee has a way to be compiled with libpurple support, but it's a bit weird to use when you have the same contact or same names present on multiple networks.

The guides below are made for Debian and it's derivatives, like Ubuntu and Mint. In order to build any of the plugins below, some common build tools are needed, apart from the per plugin specific ones:

```bash
sudo apt install libprotobuf-dev protobuf-compiler build-essential
sudo apt-get build-dep pidgin
```

How to conect to Skype with Pidgin (or libpurple)

The current iteration of the Skype plugin uses the web interface to connect to the system. It doesn't offer voice and video calls, but it supports individual and group chats alike.

If you have 2FA on, you'll need to use your app password as password and tick the `Use alternative login method` on the `Advanced` tab when adding the account.

```bash
git clone https://github.com/EionRobb/Skype4pidgin
cd Skype4pidgin/Skypeweb
cmake .
make
sudo make install
```

How to connect to Google Hangouts with Pidgin (or libpurple)

I've taken the instructions from the author's bitbucket site[^26]:

```bash
sudo apt install -y libpurple-dev libjson-glib-dev
libglib2.0-dev libprotobuf-c-dev protobuf-c-compiler
mercurial make
hg clone https://bitbucket.org/EionRobb/purple-hangouts/
cd purple-hangouts
make
sudo make install
```

How to connec to Facebook and/or Workplace by Facebook with Pidgin (or libpurple)

The Workplace support is not yet merged into the main code: it's in the `wip-work-chat` branch. More information in the support ticket[^27].

Workplace and it's 'buddy' list is sort of a mystery at this point in time, so don't expect everything to run completely smooth, but it's much better, than nothing.

In order to log in to a Workplace account, tick `Login as Workplace account` on the `Advanced` tab.

```bash
git clone https://github.com/dequis/purple-facebook
cd purple-facebook
git checkout wip-work-chat
./autogen.sh
./configure
make
sudo make install
```

How to conect to Telegram with Pidgin (or libpurple)

The Telegram plugin works nicely, including inline images and and to end encrypted messages. Voice supports seems to be lacking unfortunately.

```bash
sudo apt install libgcrypt20-dev libwebp-dev
git clone https://github.com/majn/telegram-purple
```

```
cd telegram-purple
git submodule update --init --recursive
./configure
make
sudo make install
```

How to connect to WhatsApp with Pidgin (or libpurple)

Did I mention I hate this network? **First of all a note: WhatsApp doesn't allow 3rd party applications at all. They might ban the phone number you use for life.** This ban may be extended to Facebook with the same phone number but this has never been officially confirmed.

Apart from that it needs a lot of hacking around: the plugin is not enough, because WhatsApp doesn't tell you your password. In order to get your password, you need to fake a 'registration' from the computer.

Even if you do this, only one device will work: the other instances will get logged out, so there is no way to use WhatsApp from your phone and from your laptop. It's 2007 again, except it's mobile only instead of desktop only.

Please stop using WhatsApp and use something with a tad more openness in it; XMPP, Telegram, SIP, ICQ... basically anything.

If you're stuck with needing to communicate with stubborn and lazy people, like I am, continue reading, and install the plugin for pidgin:

```bash
sudo apt install libprotobuf-dev protobuf-compiler
git clone https://github.com/jakibaki/whatsapp-purple/
cd whatsapp-purple
make
sudo make install
```

However, this is not enough: the next step is `yowsup`, a command line python utility that allows you to 'register' to WhatsApp and reveals that so well hidden password.

```bash
sudo pip3 install yowsup
```

Once done, you need to first request an SMS, meaning you'll need a number that's able to receive SMS. Replace the `COUNTRYCODE` and `PHONENUMBER` string with your country code and phone number without prefixes, so for United Kingdom, that would be:

- country code: 44
- phone number: 441234567890

No 00, or + before the full international phone number.

```bash
 $ yowsup-cli registration --requestcode sms -p
PHONENUMBER --cc COUNTRYCODE --env android

    yowsup-cli  v2.0.15
    yowsup      v2.5.7

    Copyright (c) 2012-2016 Tarek Galal
    http://www.openwhatsapp.org

    This software is provided free of charge. Copying and
redistribution is
    encouraged.

    If you appreciate this software and you would like to
support future
    development please consider donating:
    http://openwhatsapp.org/yowsup/donate



INFO:yowsup.common.http.warequest:b'{"login":"PHONENUMBER","s
tatus":"sent","length":6,"method":"sms","retry_after":
78,"sms_wait":78,"voice_wait":65}\n'
    status: b'sent'
    length: 6
    method: b'sms'
    retry_after: 78
    login: b'PHONENUMBER'
```

Once you got the SMS, use the secret code:

```bash
$ yowsup-cli registration --register SECRET-CODE -p
PHONENUMBER --cc COUNTRYCODE --env android

    yowsup-cli  v2.0.15
    yowsup      v2.5.7

    Copyright (c) 2012-2016 Tarek Galal
    http://www.openwhatsapp.org

    This software is provided free of charge. Copying and
redistribution is
    encouraged.

    If you appreciate this software and you would like to
support future
    development please consider donating:
    http://openwhatsapp.org/yowsup/donate


INFO:yowsup.common.http.warequest:b'{"status":"ok","login":"P
HONENUMBER","type":"existing","edge_routing_info":"CAA=","cha
t_dns_domain":"sl","pw":"[YOUR WHATSAPP PASSWORD YOU NEED TO
COPY]=","expiration":
4444444444.0,"kind":"free","price":"$0.99","cost":"0.99","cur
rency":"USD","price_expiration":1520591114}\n'
    status: b'ok'
    login: b'PHONENUMBER'
    pw: b'YOUR WHATSAPP PASSWORD YOU NEED TO COPY'
    type: b'existing'
    expiration: 4444444444.0
    kind: b'free'
    price: b'$0.99'
    cost: b'0.99'
    currency: b'USD'
    price_expiration: 1520591114
```

That `YOUR WHATSAPP PASSWORD YOU NEED TO COPY` is the password you need to put in the password field of the account; the username is your `PHONENUMBER`.

How to connect to WeChat with Pidgin (or libpurple)

If there is something worse, than WhatsApp, it's WeChat: app only and rather agressive when it comes to accessing private data on the phone. If you want to use it, but avoid actually serving data to it, I recommend getting the Xposed Framework[^28] with XPrivacyLua[^29] on your phone before WeChat and restricting WeChat with it as much as possible.

```bash
sudo apt install cargo clang
git clone https://github.com/sbwtw/pidgin-wechat
cd pidgin-wechat
cargo build
sudo cp target/debug/libwechat.so /usr/lib/purple-2/
```

Pidgin will only ask for a `username` - fill that in with you WeChat username and connect. Pidgin will soon pop up a window with a QR code - scan it with the WeChat app and follow the process on screen.

Other networks

Pidgin has a list of third party plugins[^30], but it's outdated. I've been searching for forks and networks missing from the list on Github.

Extra Plugins for Pidgin

Purple Plugin Pack

There are a few useful plugins for Pidgin that can make life simpler; the Purple Plugin Pack[^31] contains most of the ones in my list:

- Highlight
- IRC helper
- Message Splitter
- XMPP Priority
- Join/Part Hiding
- Markerline
- Message Timestamp Formats
- Nick Change Hiding
- Save Conversation Order
- Voice/Viceo Settings
- XMPP Service Discovery
- Mystatusbox (Show Statusboxes)

XMPP Message Carbons

XEP-0280 Message Carbons[^32] is an extension that allows multiple devices to receive all messages.

```bash
sudo apt install libpurple-dev libglib2.0-dev libxml2-dev
git clone https://github.com/gkdr/carbons
cd carbons
make
sudo make install
```

Once installed, open a chat or conversation that happens on the relevant server and type:

```
/carbons on
```

This will not be delivered as message but executed on the server as command. Unfortunately not all of the XMPP servers support this.

OMEMO

OMEMO[^33] is a multi-legged encryption protocol that allows encrypted messages across multiple devices. It's built-in into Conversations[^34], one of the best XMPP clients for Android - Pidgin doesn't have it by default.

```bash
sudo apt install git cmake libpurple-dev libmxml-dev
libxml2-dev libsqlite3-dev libgcrypt20-dev
git clone https://github.com/gkdr/lurch/
cd lurch
git submodule update --init --recursive
make
sudo make install
```

Message Delivery Receipts[^35]

Yet another missing by default XMPP extension, which is quite useful.

```bash
git clone https://git.assembla.com/pidgin-xmpp-
receipts.git
cd pidgin-xmpp-receipts/
make
sudo cp xmpp-receipts.so /usr/lib/purple-2/
```

Porting old logs to Pidgin

I wrote a Python script which can port some old logs into Pidgin. It can deal with unmodifies logs from:

- Trillian (v3.x)
- MSN Plus! HTML logs
- Skype (v2.x)

As for ZNC and Facebook, a lot of handywork is needed - see the comments in the script.

Requirements:

```bash
pip3 install arrow bs4
```

And the script:

```python
import os
import sqlite3
import logging
import re
import glob
import sys
import hashlib
import arrow
import argparse
from bs4 import BeautifulSoup
import csv


def logfilename(dt, nulltime=False):
    if nulltime:
        t = '000000'
    else:
        t = dt.format('HHmmss')

    return "%s.%s%s%s.txt" % (
        dt.format("YYYY-MM-DD"),
        t,
        dt.datetime.strftime("%z"),
        dt.datetime.strftime("%Z")
    )


def logappend(fpath,dt,sender,msg):
    logging.debug('appending log: %s' % (fpath))
    with open(fpath, 'at') as f:
        f.write("(%s) %s: %s\n" % (
        dt.format('YYYY-MM-DD HH:mm:ss'),
        sender,
        msg
    ))
    os.utime(fpath, (dt.timestamp, dt.timestamp))
    os.utime(os.path.dirname(fpath), (dt.timestamp,
dt.timestamp))


def logcreate(fpath,contact, dt,account,plugin):
    logging.debug('creating converted log: %s' % (fpath))
    if not os.path.exists(fpath):
        with open(fpath, 'wt') as f:
            f.write("Conversation with %s at %s on %s (%s)\n"
% (
                contact,
                dt.format('ddd dd MMM YYYY hh:mm:ss A ZZZ'),
                account,
                plugin
            ))


def do_facebook(account, logpathbase):
    plugin = 'facebook'

    # the source for message data is from a facebook export
    #
    # for the buddy loookup: the  pidgin buddy list xml
(blist.xml) has it, but
    # only after the alias was set for every facebook user by
hand
    # the file contains lines constructed:
```

```
    # UID\tDisplay Nice Name
    #
    lookupf = os.path.expanduser('~/tmp/facebook_lookup.csv')
    lookup = {}
    with open(lookupf, newline='') as csvfile:
        reader = csv.reader(csvfile, delimiter='\t')
        for row in reader:
            lookup.update({row[1]: row[0]})

    # the csv file for the messages is from the Facebook Data
export
    # converted with https://pypi.python.org/pypi/
fbchat_archive_parser
    # as: fbcap messages.htm -f csv > ~/tmp/facebook-
messages.csv
    dataf = os.path.expanduser('~/tmp/facebook-messages.csv')
    reader = csv.DictReader(open(dataf),skipinitialspace=True)
    for row in reader:
        # skip conversations for now because I don't have any
way of getting
        # the conversation id
        if ', ' in row['thread']:
            continue

        # the seconds are sometimes missing from the
timestamps
        try:
            dt = arrow.get(row.get('date'), 'YYYY-MM-
DDTHH:mmZZ')
        except:
            try:
                dt = arrow.get(row.get('date'), 'YYYY-MM-
DDTHH:mm:ssZZ')
            except:
                logging.error('failed to parse entry: %s',
row)

        dt = dt.to('UTC')
        contact = lookup.get(row.get('thread'))
        if not contact:
            continue
        msg = row.get('message')
        sender = row.get('sender')

        fpath = os.path.join(
            logpathbase,
            plugin,
            account,
            contact,
            logfilename(dt, nulltime=True)
        )

        if not os.path.isdir(os.path.dirname(fpath)):
            os.makedirs(os.path.dirname(fpath))
        logcreate(fpath, contact, dt, account, plugin)
        logappend(fpath, dt, sender, msg)


def do_zncfixed(znclogs, logpathbase, znctz):
    # I manually organised the ZNC logs into pidgin-like
    # plugin/account/contact/logfiles.log
    # structure before parsing them
    LINESPLIT = re.compile(
        r'^\[(?P<hour>[0-9]+):(?P<minute>[0-9]+):(?
P<second>[0-9]+)\]\s+'
        r'<(?P<sender>.*?)>\s+(?P<msg>.*)$'
    )
    searchin = os.path.join(
        znclogs,
        '**',
        '*.log'
    )
    logs = glob.glob(searchin, recursive=True)
    for log in logs:
        contact = os.path.basename(os.path.dirname(log))
        account =
os.path.basename(os.path.dirname(os.path.dirname(log)))
        plugin =
os.path.basename(os.path.dirname(os.path.dirname(os.path.dirn
ame(log))))
        logging.info('converting log file: %s' % (log))
        dt = arrow.get(os.path.basename(log).replace('.log',
''), 'YYYY-MM-DD')
        dt = dt.replace(tzinfo=znctz)


        if contact.startswith("#"):
            fname = "%s.chat" % (contact)
        else:
            fname = contact

        fpath = os.path.join(
            logpathbase,
            plugin,
            account,
            fname,
            logfilename(dt)
        )

        if not os.path.isdir(os.path.dirname(fpath)):
            os.makedirs(os.path.dirname(fpath))

        with open(log, 'rb') as f:
            for line in f:
                line = line.decode('utf8', 'ignore')
                match = LINESPLIT.match(line)
                if not match:
                    continue
                dt = dt.replace(
                    hour=int(match.group('hour')),
                    minute=int(match.group('minute')),
                    second=int(match.group('second'))
                )
                logcreate(fpath, contact, dt, account, plugin)
                logappend(fpath, dt, match.group('sender'),
match.group('msg'))


def do_msnplus(msgpluslogs, logpathbase, msgplustz):
    NOPAR = re.compile(r'\((.*)\)')
    NOCOLON = re.compile(r'(.*):?')

    searchin = os.path.join(
        msgpluslogs,
        '**',
        '*.html'
    )
    logs = glob.glob(searchin, recursive=True)
    plugin = 'msn'
    for log in logs:
        logging.info('converting log file: %s' % (log))
        contact = os.path.basename(os.path.dirname(log))

        with open(log, 'rt', encoding='UTF-16') as f:
            html = BeautifulSoup(f.read(), "html.parser")
            account = html.find_all('li',
attrs={'class':'in'}, limit=1)[0]
            account = NOPAR.sub('\g<1>', account.span.string)
            for session in html.findAll(attrs={'class':
'mplsession'}):
                dt = arrow.get(
                    session.get('id').replace('Session_', ''),
                    'YYYY-MM-DDTHH-mm-ss'
                )
                dt = dt.replace(tzinfo=msgplustz)
                seconds = int(dt.format('s'))

                fpath = os.path.join(
                    logpathbase,
                    plugin,
                    account,
                    contact,
                    logfilename(dt)
                )

                if not os.path.isdir(os.path.dirname(fpath)):
                    os.makedirs(os.path.dirname(fpath))

                for line in session.findAll('tr'):
                    if seconds == 59:
                        seconds = 0
                    else:
                        seconds = seconds + 1
                    tspan = line.find(attrs={'class':
'time'}).extract()
                    time = tspan.string.replace('(',
'').replace(')','').strip().split(':')

                    sender = line.find('th').string
                    if not sender:
                        continue

                    sender = sender.strip().split(':')[0]
                    msg = line.find('td').get_text()

                    mindt = dt.replace(
                        hour=int(time[0]),
                        minute=int(time[1]),
                        second=int(seconds)
                    )

                    logcreate(fpath, contact, dt, account,
plugin)
                    logappend(fpath, mindt, sender, msg)


def do_trillian(trillianlogs, logpathbase, trilliantz):
    SPLIT_SESSIONS = re.compile(
        r'^Session Start\s+\((?P<participants>.*)?\):\s+(?
P<timestamp>[^\n]+)'
        r'\n(?P<session>(?:.|\n)*?)(?=Session)',
        re.MULTILINE
    )

    SPLIT_MESSAGES = re.compile(
        r'\[(?P<time>[^\]]+)\]\s+(?P<sender>.*?):\s+'
        r'(?P<msg>(?:.|\n)*?)(?=\n\[|$)'
    )

    searchin = os.path.join(
        trillianlogs,
        '**',
        '*.log'
    )
    logs = glob.glob(searchin, recursive=True)
    for log in logs:
        if 'Channel' in log:
            logging.warn(
                "Group conversations are not supported yet,
skipping %s" % log
            )
            continue

        logging.info('converting log file: %s' % (log))
        contact = os.path.basename(log).replace('.log', '')
        plugin =
os.path.basename(os.path.dirname(os.path.dirname(log))).lower
()

        with open(log, 'rb') as f:
            c = f.read().decode('utf8', 'ignore')

            for session in SPLIT_SESSIONS.findall(c):
                participants, timestamp, session = session
                logging.debug('converting session starting
at: %s' % (timestamp))
                participants = participants.split(':')
                account = participants[0]
                dt = arrow.get(timestamp, 'ddd MMM DD
HH:mm:ss YYYY')
                dt = dt.replace(tzinfo=trilliantz)
                fpath = os.path.join(
                    logpathbase,
                    plugin,
                    participants[0],
                    contact,
                    logfilename(dt)
                )

                if not os.path.isdir(os.path.dirname(fpath)):
                    os.makedirs(os.path.dirname(fpath))

                seconds = int(dt.format('s'))
                curr_mindt = dt
```

```
                    for line in SPLIT_MESSAGES.findall(session):
                        # this is a fix for ancient trillian logs
where seconds
                        # were missing
                        if seconds == 59:
                            seconds = 0
                        else:
                            seconds = seconds + 1

                        time, sender, msg = line
                        try:
                            mindt = arrow.get(time,
                            'YYYY.MM.DD HH:mm:ss')
                        except:
                            time = time.split(':')
                            mindt = dt.replace(
                                hour=int(time[0]),
                                minute=int(time[1]),
                                second=int(seconds)
                            )

                        # creating the filw with the header has
to be here to
                        # avoid empty or status-messages only
files
                        logcreate(fpath, participants[1], dt,
account, plugin)
                        logappend(fpath, mindt, sender, msg)

                if params.get('cleanup'):
                    print('deleting old log: %s' % (log))
                    os.unlink(log)


def do_skype(skypedbpath, logpathbase):
    db = sqlite3.connect(skypedbpath)

    cursor = db.cursor()
    cursor.execute('''SELECT `skypename` from Accounts''')
    accounts = cursor.fetchall()
    for account in accounts:
            account = account[0]
            cursor.execute('''
            SELECT
                `timestamp`,
                `dialog_partner`,
                `author`,
                `from_dispname`,
                `body_xml`
            FROM
                `Messages`
            WHERE
                `chatname` LIKE ?
            ORDER BY
                `timestamp` ASC
            ''', ('%' + account + '%',))

            messages = cursor.fetchall()
            for r in messages:
                dt = arrow.get(r[0])
                dt = dt.replace(tzinfo='UTC')
                fpath = os.path.join(
                    logpathbase,
                    account,
                    r[1],
                    logfilename(dt, nulltime=True)
                )

                if not os.path.isdir(os.path.dirname(fpath)):
                    os.makedirs(os.path.dirname(fpath))

                logcreate(fpath, r[1], dt, account, 'skype')
                logappend(fpath, dt, r[3], r[4])


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Parameters
for Skype v2 logs to Pidgin logs converter')

    parser.add_argument(
            '--skype_db',
            default=os.path.expanduser('~/.skype/main.db'),
            help='absolute path to skype main.db'
        )

        parser.add_argument(
            '--pidgin_logs',
            default=os.path.expanduser('~/.purple/logs/skype'),
            help='absolute path to Pidgin skype logs'
        )

        parser.add_argument(
            '--facebook_account',
            default='',
            help='facebook account name'
        )

        parser.add_argument(
            '--loglevel',
            default='warning',
            help='change loglevel'
        )

        for allowed in ['skype', 'trillian', 'msnplus', 'znc',
'facebook']:
            parser.add_argument(
                '--%s' % allowed,
                action='store_true',
                default=False,
                help='convert %s logs' % allowed
            )

            if allowed != 'skype' or allowed != 'facebook':
                parser.add_argument(
                    '--%s_logs' % allowed,
                    default=os.path.expanduser('~/.%s/logs' %
allowed),
                    help='absolute path to %s logs' % allowed
                )

                parser.add_argument(
                    '--%s_timezone' % allowed,
                    default='UTC',
                    help='timezone name for %s logs (eg. US/
    Pacific)' % allowed
                )

        params = vars(parser.parse_args())

        # remove the rest of the potential loggers
        while len(logging.root.handlers) > 0:
            logging.root.removeHandler(logging.root.handlers[-1])

        LLEVEL = {
            'critical': 50,
            'error': 40,
            'warning': 30,
            'info': 20,
            'debug': 10
        }

        logging.basicConfig(
            level=LLEVEL[params.get('loglevel')],
            format='%(asctime)s - %(levelname)s - %(message)s'
        )

        if params.get('facebook'):
            logging.info('facebook enabled')
            do_facebook(
                params.get('facebook_account'),
                params.get('pidgin_logs')
            )


        if params.get('skype'):
            logging.info('Skype enabled; parsing skype logs')
            do_skype(
                params.get('skype_db'),
                params.get('pidgin_logs')
            )

        if params.get('trillian'):
            logging.info('Trillian enabled; parsing trillian
    logs')
```

```
        do_trillian(
            params.get('trillian_logs'),
            params.get('pidgin_logs'),
            params.get('trillian_timezone'),
        )

    if params.get('msnplus'):
        logging.info('MSN Plus! enabled; parsing logs')
        do_msnplus(
            params.get('msnplus_logs'),
            params.get('pidgin_logs'),
            params.get('msnplus_timezone'),
        )

    if params.get('znc'):
        logging.info('ZNC enabled; parsing znc logs')
        do_zncfixed(
            params.get('znc_logs'),
            params.get('pidgin_logs'),
            params.get('znc_timezone'),
        )
```

Links

1. http://www.irc.org/
2. https://www.mirc.com/
3. https://icq.com/
4. https://medium.com/@Dimitryophoto/icq-20-years-is-no-limit-8734e1eea8ea
5. https://en.wikipedia.org/wiki/Windows_Live_Messenger
6. https://www.trillian.im/
7. http://pidgin.im/
8. https://xmpp.org/
9. https://xmpp.org/extensions/xep-0280.html
10. https://en.wikipedia.org/wiki/Whatsapp
11. https://en.wikipedia.org/wiki/BlackBerry_Messenger
12. https://en.wikipedia.org/wiki/Google_talk
13. https://en.wikipedia.org/wiki/Google_Hangouts
14. https://en.wikipedia.org/wiki/MQTT
15. http://www.salimvirani.com/facebook/
16. https://www.facebook.com/workplace

17. https://www.blog.google/products/g-suite/move-projects-forward-one-placehangouts-chat-now-available/
18. https://telegram.org/
19. https://xmpp.org/extensions/xep-0384.html
20. https://matrix.org/
21. https://xmpp.org/extensions/xep-0384.html
22. https://xmpp.org/extensions/xep-0166.html
23. https://play.google.com/store/apps/details?id=com.mailsite.astrachat
24. https://f-droid.org/packages/eu.siacs.conversations/
25. https://www.linphone.org/
26. https://bitbucket.org/EionRobb/purple-hangouts/src#markdown-header-compiling
27. https://github.com/dequis/purple-facebook/issues/371
28. http://repo.xposed.info/
29. https://lua.xprivacy.eu/
30. https://developer.pidgin.im/wiki/ThirdPartyPlugins
31. https://bitbucket.org/rekkanoryo/purple-plugin-pack/
32. https://xmpp.org/extensions/xep-0280.html
33. https://xmpp.org/extensions/xep-0384.html
34. https://f-droid.org/packages/eu.siacs.conversations/
35. https://xmpp.org/extensions/xep-0184.html