bgpipe:

open source BGP reverse proxy

www.bgpipe.org

Pawel Foremski <pjf@iitis.pl>

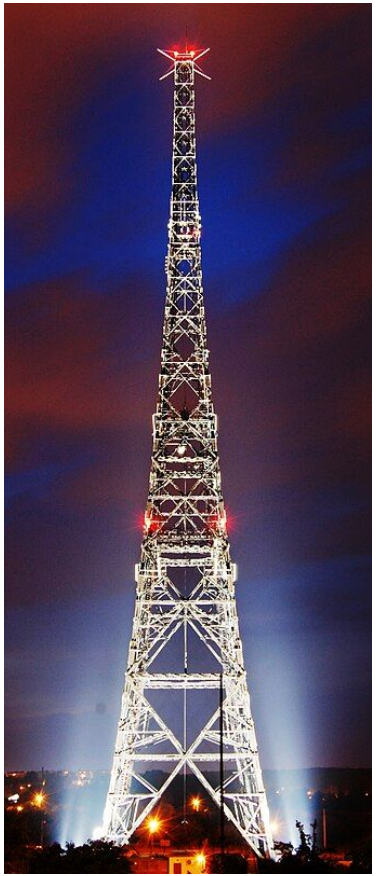
IITiS PAN / DomainTools

RIPE 88

Kraków, Poland 2024

Cześć! /chech-sh-ch/

- Assistant Professor @ IITiS PAN
- Software engineer and network admin, currently @ DomainTools
- DNS, BGP, IPv6 - eg. see RIPE 74 talk

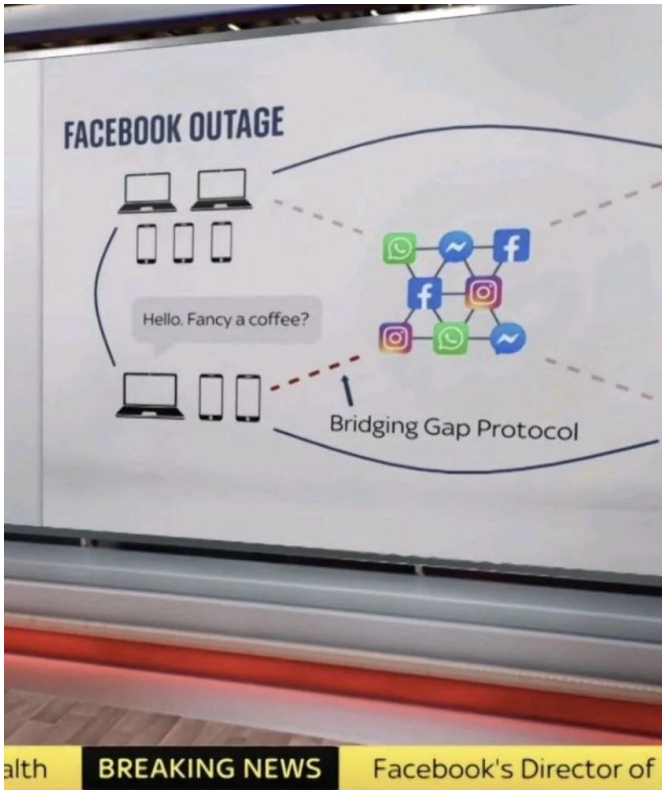


Gliwice

@pforemski

bgpipe bridging the gaps :)

- ...in BGP academic vs. real worlds
  - eg. practical fix for Kirin attack
- ...in BGP innovation
  - related: xBGP @ RIPE87
- ...in BGP big vendors vs. everyday fires
  - eg. quick fix for grave flaws in BGP error handling
- ...in BGP open source world
  - a better alternative to existing CLI tools



bgpipe

- *"BGP reverse proxy and firewall"*

- Build BGP pipelines - eg:
  - man-in-the-middle BGP proxy for session auditing, observability, archival, etc.
  - translate BGP to JSON and back
  - tunnel BGP sessions over encrypted websockets
  - implement sophisticated session limits
  - pipe your sessions through ~~sed~~ Python scripts
  - add RPKI (and ASPA) validation to old routers

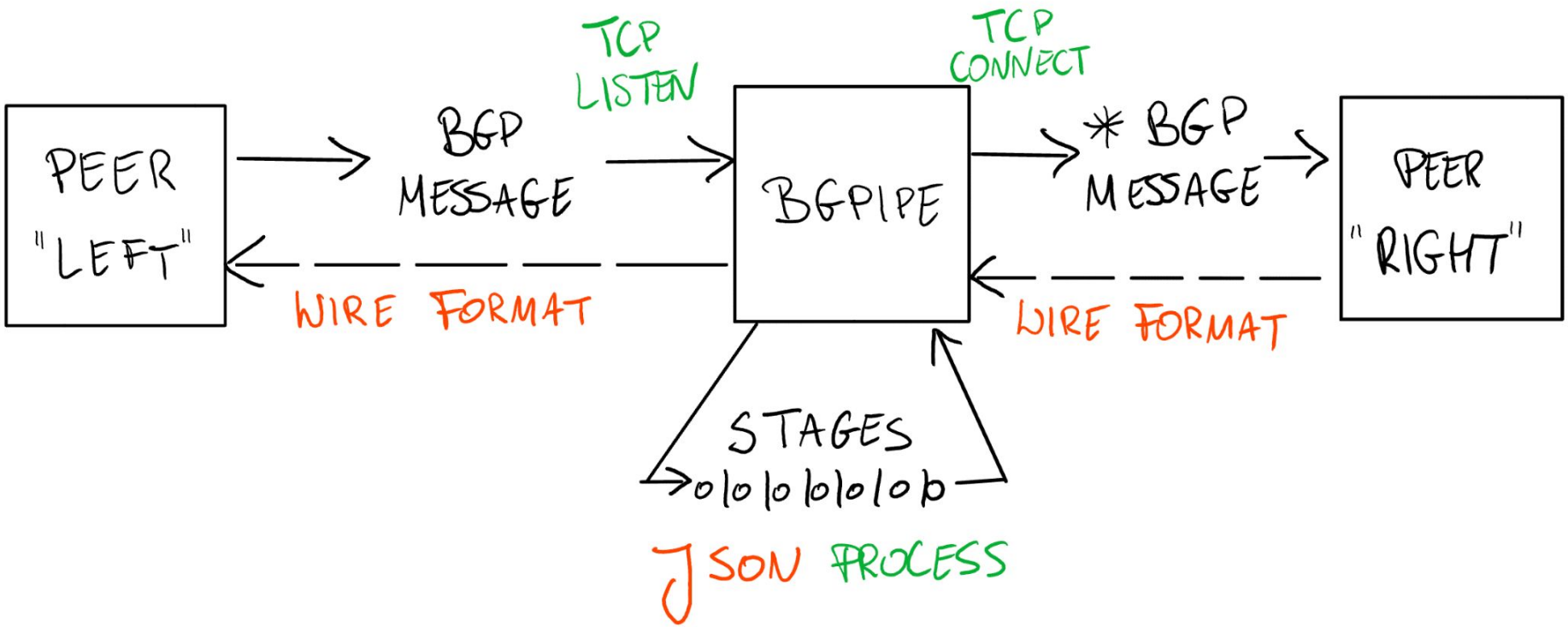
- General goal: *"universal streaming message processor for tinkering with BGP in prod"*

- Main parts: BGPFix.org + bgpipe.org
- Started early 2023
- Implemented in Golang (GC-aware)
- Licensed under the MIT license
- Third-party plugins independent (use API locally or remotely)

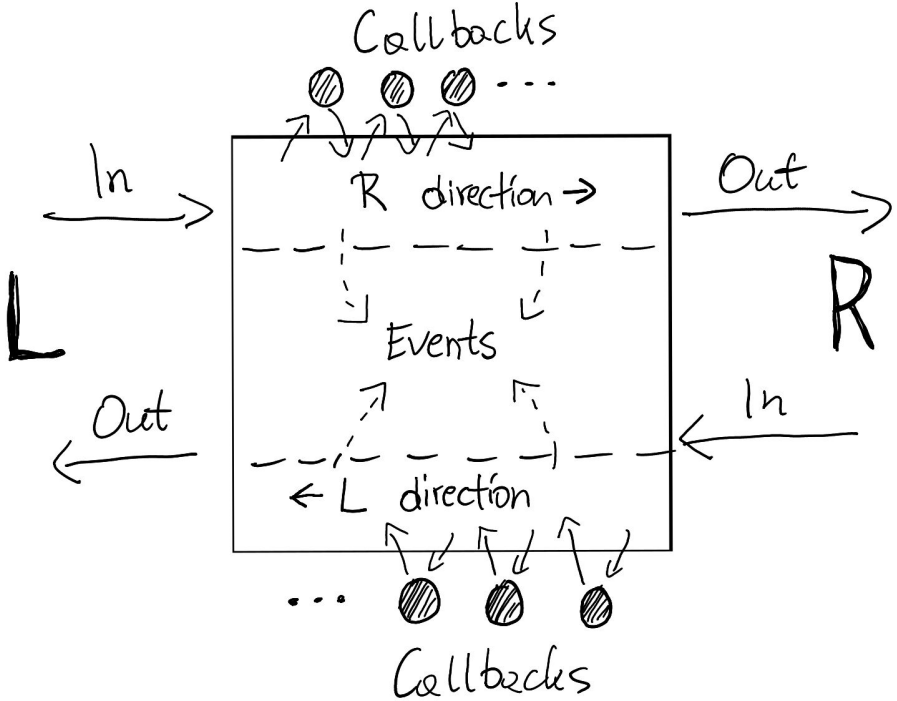
- Status: ready for lab R&D

- Next step: early adopters feedback

Overview


TCP
LISTEN
TCP
CONNECT
PEER
"LEFT"
BGP
MESSAGE
BGPIPE
* BGP
MESSAGE
PEER
"RIGHT"
WIRE FORMAT
WIRE FORMAT
STAGES
JSON PROCESS

BGPFix backend



- Pipeline: collection of Stages (ordered)
- Stage: self-contained “plugin”, can work anywhere in Pipeline (L/R direction or both). Uses Callbacks and Events.
- Callback: a function to call for matching BGP messages. Can modify / drop / create BGP messages, emit Events, etc.
- Event: information broadcast to all Stages, which can subscribe to particular Events. Usually references a BGP message.

Stages & the CLI

- TCP/IP
  - connect
  - listen
- File I/O
  - stdin
  - stdout
  - read
  - write
- Filtering
  - exec
  - pipe
  - websocket
- BGP
  - speaker
  - limit

```
pjf@pjf:~/bgpfix/bgpipe$ ./bgpipe -h
Usage: bgpipe [OPTIONS] [--] STAGE1 [OPTIONS] [ARGUMENTS] [--] STAGE2...

Options:
  -v, --version          print detailed version info and quit
  -l, --log string       log level (debug/info/warn/error/disabled) (defa
  -e, --events strings   log given events ("all" means all events) (defau
  -k, --kill strings     kill session on any of these events
  -i, --stdin            read JSON from stdin
  -o, --stdout           write JSON to stdout
  -I, --stdin-wait       like --stdin but wait for EVENT_ESTABLISHED
  -O, --stdout-wait      like --stdout but wait for EVENT_EOR
  -2, --short-asn        use 2-byte ASN numbers
```

```
# quick session with 192.168.1.1, dump to JSON
$ bgpipe -o speaker 192.168.1.1

# proxy 1.2.3.4 to 10.1.0.1, adding TCP-MD5
$ bgpipe \
      -- listen 1.2.3.4 \
      -- connect --md5 "solarwinds123" 10.1.0.1

# sed-in-the-middle proxy rewriting ASNs in OPEN messages
$ bgpipe \
      -- connect 127.0.0.1 \
      -- exec -LR --args sed -ure '/"OPEN"/{ s/65055/65001/g; }' \
      -- connect 85.232.240.179
```

PoC: remote message archival over websocket

```
# on edge
$ bgpipe \
  -- speaker --asn 65000 \
  -- websocket -L --write wss://isp.com \
  -- connect 192.168.1.1


# on the archival host
$ bgpipe \
  -- websocket -R --read --listen wss://isp.com \
  -- write --every 15m --compress \
     '/var/bgpipe/bgp.$TIME.json.gz'
```

```
pjf@pjf:~/bgpfix/bgpipe$ ./bgpipe websocket -h
Stage usage: websocket [OPTIONS] URL

Description: filter messages over websocket

Options:
      --listen              listen on given URL instead of dialing it
      --auth string         use HTTP basic auth ($ENV_VARIABLE or file p
      --cert string         SSL certificate path
      --key string          SSL private key path
      --insecure            do not verify the SSL certificate
      --header strings      HTTP headers to send in client mode
      --timeout duration    connect timeout (0 means none) (default 10s)
      --raw                 speak raw BGP instead of JSON
      --mrt                 speak MRT-BGP4MP instead of JSON
      --type strings        skip if message is not of specified type(s)
      --read                read-only mode (no output from bgpipe)
      --write               write-only mode (no input to bgpipe)
      --copy                copy messages instead of filtering (mirror)
      --pardon              ignore input parse errors
      --no-seq              overwrite input message sequence number
      --no-time             overwrite input message time
      --no-tags             drop input message tags
```

PoC: adding RPKI validation with Routinator

validator.py:

```
1 import sys, lib, json
2 for line in sys.stdin:
3   try:
4     msg = lib.parse_json(line)
5
6     # validate prefixes vs. the origin AS
7     origin = int(msg[5]["attrs"]["ASPATH"]["value"][-1])
8     ok, fail = [], []
9     for pfx in msg[5]["reach"]:
10      ok.append(pfx) if lib.rpki_check(origin, pfx) \
11        else fail.append(pfx)
12
13    # treat invalid prefixes as withdrawn, like rfc7606
14    if len(fail) > 0:
15      msg[5]["reach"] = ok
16      msg[5]["unreach"].extend(fail)
17      print(json.dumps(msg), flush=True) # rewrite msg
18    else:
19      raise # accept as-is
20  except:
21    print(line, end="", flush=True)
```

```
# pipe BGP session through validator.py
$ bgpipe 192.168.1.1 -- exec -R ./validator.py -- 1.2.3.4
```

lib.py

```
1 import sys, json, requests
2
3 def parse_json(line: str) -> object:
4     msg = json.loads(line)
5     if msg[4] != "UPDATE" or len(msg[5]["reach"]) == 0:
6         raise Exception
7     if "unreach" not in msg[5]:
8         msg[5]["unreach"] = list() # in case it's needed
9     return msg
10
11 def rpki_check(origin: int, prefix: str) -> bool:
12     URL = "http://127.0.0.1:31339/api/v1/validity"
13     try:
14         url = f"{URL}/{origin}/{prefix}"
15         response = requests.get(url)
16         response.raise_for_status()
17         result = response.json()
18         return result ["validated_route"] \
19             ["validity"]["state"] != "invalid"
20     except Exception as err:
21         print(err, file=sys.stderr)
22         raise
```

PoC: stopping Kirin

- Context: pub.foremski.pl/2024-kirin.pdf

Kirin: Hitting the Internet with Distributed BGP Announcements

Lars Prehn
MPI-INF
lprehn@mpi-inf.mpg.de

Pawel Foremski
IITiS PAN / DomainTools
pjf@iitis.pl

Oliver Gasser
IPinfo / MPI-INF
oliver@ipinfo.io

- More advanced max-prefix limits:
  - per-session (classic)
  - per-IP block (eg. 10k per each ::/32)
  - per-AS origin (eg. 15k for any ASN)

- Implemented as a Stage: see [limit.go]

```
pjf@pjf:~/bgpfix/bgpipe$ ./bgpipe limit -h
Stage usage: limit [OPTIONS]

Description: limit prefix lengths and counts

Options:
  -4, --ipv4                process IPv4 prefixes
  -6, --ipv6                process IPv6 prefixes
      --multicast           process multicast prefixes
      --permanent           make announcements permanent (do not con
  -m, --min-length int      min. prefix length (0 = no limit)
  -M, --max-length int      max. prefix length (0 = no limit)
  -s, --session int         global session limit (0 = no limit)
  -o, --origin int          per-AS origin limit (0 = no limit)
  -b, --block int           per-IP block limit (0 = no limit)
  -B, --block-length int    IP block length (max. 64, 0 = 8/32 for v

Common Options:
  -L, --left                operate in the L direction
  -R, --right               operate in the R direction
  -A, --args                consume all CLI arguments till --
  -W, --wait strings        wait for given event before starting
  -S, --stop strings        stop after given event is handled

Events:
  limit/block               too many prefixes for a single IP block
  limit/count               too many prefixes reachable over the ses
  limit/long                too long prefix announced
  limit/origin              too many prefixes for a single AS origin
  limit/short               too short prefix announced
```

Take-away

Start using today! ✅

- bgpipe is easy to start inspecting & hacking your BGP sessions

Goals: 🏆

- now: looking for early adopters & feedback
- mid-term: collaboration (coding / research)
- long-term: stable & sustainable project

bgpipe.org

```
$ go install github.com/bgpfix/bgpipe@latest
```

Pawel Foremski <pjf@iitis.pl>

@pforemski