Lwresd:
how can be obsolete daemon reused for new features?

Petr Menšík <pemensik@redhat.com>

Red Hat

FOSDEM 2026

Who am I?

- Linux user since ~2000, started with Monkey Linux
- Debian user during study at FIT University of Technology, Brno
- Red Hatter since 2016, switched to Fedora, where I maintain packages in work and free time
- Avahi upstream maintainer for ~2 years

Motivation

- **Typical application does not use DNS directly**
- **Applications should use getaddrinfo() library call**
- **Applications specify what address family they want**
  - Defaults to AF_UNSPEC, AF_INET and AF_INET6 are other variants
  - Required for link-local IPv6 addresses with interface index
    - ping -c3 fe80::105a:412a:503e:8660%enp0s20f0u14
  - ping -4, curl -6, dig -4 changes only used AF

What can I pass into getaddrinfo?

```
/* man 3 getaddrinfo */
#include <netdb.h>

int getaddrinfo(const char *restrict node,
                const char *restrict service,
                const struct addrinfo *restrict hints,
                struct addrinfo **restrict res);
```

// added by ipv6 wg RFC 3493:
https://www.rfc-editor.org/rfc/rfc3493.html#section-6

What can I pass in struct addrinfo?

```
struct addrinfo {
    int             ai_flags;     // AI_NUMERICHOST, AI_PASSIVE, ...
    int             ai_family;  // AF_UNSPEC, AF_INET, AF_INET6
    int             ai_socktype; // SOCK_STREAM or SOCK_DGRAM
    int             ai_protocol;  // usually 0?
    socklen_t       ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname; // ~ final host name
    struct addrinfo *ai_next;  // can be more than 1 address
};
```

Motivation #2

- **Why does my IPv4-only network generate so many AAAA requests?**
  - Why does it generate both A and AAAA query on each AF_UNSPEC request?
    - Supressing AAAA? on IPv4-only networks is done already on Windows and MacOS
    - Filtering on DNS cache does not receive original address family requested
  - The same applies for IPv6-only networks, not a legacy-only problem
  - Fake empty AAAA responses break DNSSEC

Motivation #3

- **What if my machine asked only queries it needs?**
  - connect() to any not localhost or link-local IPv6 address will always fail, unless there is at least some route
    - Systemd-resolved had it controlled by default route, but they removed the functionality
    - Proposed change in glibc to use resolv.conf options *ipv4*, *ipv6* (https://sourceware.org/bugzilla/show_bug.cgi?id=30544)
    - Not asking will not break caches or DNSSEC validation
    - On mobile devices dynamic network changes might need fast reaction

Motivation #4

- **Dynamic changes will work best when handled by a common localhost service**
- **getaddrinfo() calls are stateless, no autodiscovered state remain for requests made later**
  - option edns0 – we could autodetect support after 1st response
  - Multi-qtype support needs the same kind of autodetection
    - Proposed for SRV+TXT queries, used by DNS-SD queries
    - The same support would be useful for A+AAAA+HTTPS queries done by any web client
    - https://datatracker.ietf.org/doc/draft-ietf-dnssd-multi-qtypes/

What can we use to cache getaddrinfo calls?

- **nscd exists, never had a port domain network socket**
  - But obsoleted by glibc already
- **systemd-resolved has also resolve plugin**
  - Uses unix domain socket by own protocol
  - But pushing them to fix some bugs is ***very*** difficult
- **lwres had also own nss plugin in Debian 3.0**
  - Its sources are not in Debian anymore, found them in Ubuntu!

Missing parameters

- **Glibc plugin interface for getaddrinfo lack address family and ai_flags!**
  - Without new glibc interface we cannot make some decisions
    - No direct address family
    - No ai_flags passed into plugin function
    - AI_PASSIVE flag may affect what addresses we provide

What can nss plugin implement?

```
enum nss_status _nss_gethostbyname4_r(
                                const char* name,
                                struct gaih_addrtuple** pat,
                                char* buffer, size_t buflen,
                                int* errnop, int* h_errnop,
                                int32_t* ttlp);

/* used by getent ahosts example.org */
```

gethostbyname4_r() parameters

```
/* Data structure used for the 'gethostbyname4_r' function.  */
struct gaih_addrtuple
  {
    struct gaih_addrtuple *next;
    char *name;
    int family;
    uint32_t addr[4];
    uint32_t scopeid;
  };
```

What is needed then?

- **Send requests from stateless application to caching daemon as specified by API calls**
- **Use unix-socket for communication with local-only service**
- **Local sockets have separate namespace!**
  - Separate users can have own instance not colliding with main system daemon!
  - Users can have own customized instances of name resolver

What did libnss_lwres implement?

- Version 0.93
- enum nss_status
  _nss_lwres_gethostbyname2_r (const char *name, int af,
  struct hostent *result,
  char *buffer, size_t buflen, int *errnop,
  int *herrnop);
- lwres_getipnodebyname (name, af, mapped_flags, herrnop);

What did lwresd implement?

- **Bin 9.11 still had unix domain sockets support!**
- ***lwresd* never used it for queries**
- **Iteration from built-in root servers hints supported**
- **It was ~ special *named* service**
  - Listening on custom binary protocol, localhost UDP port 921
  - I still maintain it on RHEL8!
  - BIND 9.11 ARM, supports views

What it should implement?

- No root hints iteration – protective DNS should see all queries
- Small default cache size
- No crypto stuff – DNSSEC validation should be handled by DNS proxy, separate service
- Unix domain port listening in /run, SOCK_STREAM
- Smart defaults, no configuration needed

Why unix domain sockets?

- struct sockaddr_un {

  sa_family_t sun_family; /* AF_UNIX */

  char sun_path[108]; /* Pathname */

  };
- getsockopt SO_PEERCRED, SO_PEERSEC
- https://github.com/avahi/avahi/pull/808

Why unix domain sockets? #2

- struct ucred {

  pid_t pid; /* Process ID of the sending process */

  uid_t uid; /* User ID of the sending process */

  gid_t gid; /* Group ID of the sending process */

  };
- man 7 unix, Linux specific only!

Why unix domain sockets? #3

- FreeBSD: LOCAL_PEERCRED
- struct xucred {

```
    u_int cr_version;               /* structure layout version */
    uid_t cr_uid;                   /* effective user id */
    short cr_ngroups;               /* number of groups */
    gid_t cr_groups[XU_NGROUPS];  /* groups */
   pid_t cr_pid;                    /* process id of the sending process */
};
```

- man 4 unix, FreeBSD specific only!
- Different *struct cmsgcred* for SOCK_DGRAM

How to log unix domain socket request?

- **struct sockaddr_un is the same as server's, not interesting**
- **Uid is more useful, but readable user name needs getpwent_r()**
- **Pid allows fetching more details, but cannot be atomic**
  - Will work best if details are fetched before the response is sent
  - /proc/$pid/cmdline
  - /proc/$pid/cgroup (Linux specific again?)
    - Can use sd_pid_get_cgroup, sd_pid_get_slice or sd_pid_get_unit from libsystemd

What could it provide?

- **Optional caching even before nss_dns plugin (/etc/resolv.conf)**
- **User owned ~/.config/hosts file**
  - Even better ~/.config/hosts.d/*.host
  - Dynamic reconfiguration from other services events
  - Automatic /etc/resolv.conf change monitoring
- **Filtering internet access similar to SELinux on filesystem**
- **Caching for LLMNR or mDNS requests too**

Red Hat

Questions?

Petr Menšík
Mail:
pemensik@redhat.com
Mattermost:
pemensik@chat.dns-oarc.net
Matrix: pemensik:fedora.im