Why resolving two names in a GUI program

is hard

•••

Summary of available name resolution APIs on Linux and why a new one is needed

How can app resolve names?

getaddrinfo(3)

- Address family and protocol independent
- Requires just hostname and service name
- Returns ordered list of address with mixed AF
- Supported on all major OS
- Resolution protocol independent
- ***Blocks* thread until finished**

DNS only libraries

- Some provide also asynchronous resolution
  - getdns
  - unbound library
  - adns
  - c-ares
- Won’t resolve other protocol names at all
- Limits mobile devices or workstations, not servers

Not only DNS provides name resolution

- getaddrinfo() on GNU/Linux serve names from configurable NSS plugins
- Name Service Switch can use different modules
  - files – local */etc/hosts* file with hostname overrides
  - MDNS – local LAN name resolution over Bonjour (nss-mdns)
  - LLMNR – local LAN resolution present on Windows (enabled in systemd-resolved)
  - WINS – Netbios based resolution from Samba (samba-winbind-modules, obsolete?)
  - Libvirt – Virtual machines running on this host (nss-libvirt)
  - DNS – usually tried last
- Common application should use names provided by any of them

Systemd-resolved APIs

- Provides DBus resolution API and port 53 stub
- But no other service provides compatible interface
- Supports multiple protocols
- Breaks DNS-only applications
  - Forwards DNS queries only to non-DNS protocols
  - Causes own kind of regressions (#23622, #23737)

How can I make multiple connections?

BSD (and Linux) socket(7) interface

- Can work with both streamed TCP and datagram UDP
- Present on most operating systems with small differences
- Even single thread can handle dozens of connections!
- Use **poll(2)** or **select(2)** to process only sockets with received data

Is blocking a problem?

Graphical application requirements

- Blocking call in the main thread makes application non-responsive
- Every GUI application can handle multiple sockets
- Input events from are delivered over (some) socket
  - From other applications or services too
- Applications implements just callbacks to events
- Spends most of time waiting for events

Just spawn a thread, right?

Correct work with threads is difficult

- Spawning a new thread is simple
- Receiving its results in the main thread is not
- Thread communication increases complexity

Why do we need a thread anyway?

What does name resolution?

- Obtain answer from fast local storage
  - files, libvirt – read some data from disk
- Ask some service on local or remote host and wait for answer
  - Use some socket(s) to send request(s)
  - May *wait noticeable* period of time
  - Extract addresses from protocol-specific response and return them to the caller
  - mdns, resolve, wins
- Waiting for *timeout or socket* activity is implemented by most frameworks

How can it be made non-blocking?

- Use common code to implement protocol-specific plugins
- Provide a way to work in custom event loops
  - Not only Qt and GLib are used in applications
- Rewrite existing NSS modules to use callbacks instead of blocking
  - Current NSS modules are easy to write, but difficult to use
  - Resolution should be simple even in non-trivial applications
- Eventloop integration module has to offer:
  - Ability to add/modify sockets to the watched list and specify events to watch
  - Be notified after some time elapsed without any socket activity (timeout handling)
  - Provide callbacks to handle socket events and timeout events
  - Time precision requirement is not important (timeouts are often in seconds)

Why non-blocking?

- Queries do not communicate between threads – no race conditions
- Query number limited only by the number of sockets and timers handled
  - Almost unlimited usually
  - Much cheaper than thread per query
- Single connection can stay in a single thread
  - Resolution becomes more similar to network data processing
  - Worker threads still make sense sometime
    - Small JSON data × Disk intensive jobs
- Server software could use simplified resolution too

I like it, where is the implementation?

- No working code yet :-(
- The most similar implementation
  - GitHub - crossdistro/netresolve – written by Pavel Šimerda
  - Implements separate loadable modules
  - But non-blocking API is missing
  - Documentation is poor
- I would like to extend it, but first need feedback
- If we add metadata parameters array to *struct addrinfo*, it may work also for HTTPS RR
  - At least SRV is used in both DNS and Multicast DNS for similar thing

Questions?

Contacts

- Email: Petr Menšík <pemensik@redhat.com>
- Matrix: @pemensik:fedora.im
- IRC: libera.chat, pemensik at #dns
- GitHub: https://github.com/pemensik
- GitLab: https://gitlab.com/pemensik