



Peer-to-Peer Systems and Security

Introduction to GUNet 0.9.x for Developers

Christian Grothoff

Lehrstuhl für Netzarchitekturen und Netzdienste
Institut für Informatik
Technische Universität München

April 20, 2012



Agenda

- GUNet 0.9.x Release Status
- GUNet 0.9.x Features
- GUNet 0.9.x System Overview
- GUNet 0.9.x APIs



What is GNUnet?

- GNU software package with 350k+ LOC in C
- P2P framework with focus on “security”
- Research project with over 20 related publications



Applications built using GNUnet

- Anonymous and non-anonymous file-sharing
- IPv6–IPv4 protocol translator and tunnel (P2P-based IPv6 migration)
- “The GNUnet Naming System”, a censorship-resistant replacement for DNS
- SNAG “social networking application for GNUnet” (external development)
- ...



GNUnet 0.9.x Release Status

- GNUnet 0.9.2 is an alpha release
- GNUnet 0.9.2 works on GNU/Linux, OS X, likely Solaris
- GNUnet 0.9.2 has known bugs (see TODO, Mantis)
- GNUnet 0.9.2 lacks documentation
- GNUnet 0.9.2 has a somewhat steep learning curve

We hope to release 0.9.3 shortly with fewer bugs, better documentation, ...



Agenda

- GUNet 0.9.x Release Status
- **GUNet 0.9.x Features**
- GUNet 0.9.x System Overview
- GUNet 0.9.x APIs



GNUnet 0.9.x Features

- OS abstraction layer
- Bandwidth management
- Transport abstraction (TCP, UDP, ...)
- Link encryption
- Peer discovery (hostlist, P2P gossip)
- Topology management



GNUnet 0.9.x Features

- Logging, configuration management, command-line parsing
- Cryptographic primitives
- Event loop, client-server IPC messaging infrastructure
- Binary I/O, asynchronous DNS resolution,
- Datastructures (Heap, HashMap, Bloomfilter)



GNUnet 0.9.x Features

- Datastore (for file-sharing)
- Datacache (for DHT)
- Statistics
- Network Size Estimation
- Testbed management (loopback & distributed testing)
- Automatic Restart Management



GNUnet 0.9.x DHT Features

- Randomized DHT based on Kademlia
- Command-line interface (GET/PUT)
- Client-library (C API)
- Should work pretty well, but unreliable as any P2P operation (!)



GNUnet 0.9.x Experimental Features

- Multicast mesh routing
- TCP-like stream abstraction over mesh
- Helper libraries for IP, DNS, Postgres and MySQL interactions
- QoS support (“ats” performance API)



Agenda

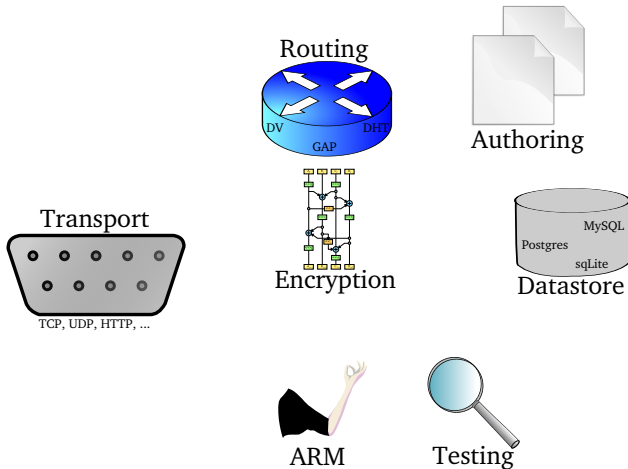
- GUNet 0.9.x Release Status
- GUNet 0.9.x Features
- **GUNet 0.9.x System Overview**
- GUNet 0.9.x APIs



- <https://gnunet.org/>
 - How to build & run GNUnet
 - End-user and developer manuals, FAQ
 - Bug database
 - Doxygen source code documentation
 - Regression tests results
 - Code coverage analysis
 - Static analysis
- <irc.freenode.net#gnunet>



GNUnet System Overview





GNUnet System Overview: 0.9.x Philosophy

- `gnunetutil` library provides shared functions for services, daemons and user interfaces
- No (more) threads (no deadlocks, no races, no fun)
- Services are processes accessed via C API
- Daemons are processes without an API
- Service API use IPC (TCP/IP or UNIX Domain Sockets) to communicate with the respective service process
- Service processes are managed by `gnunet-service-arm`
- `gnunet-service-arm` is controlled with `gnunet-arm`



- autoconf, automake, libtool, gcc
- libgmp
- libgcrypt ≥ 1.2
- libmicrohttpd $\geq 0.9.18!$
- libextractor $\geq 0.6.x!!$
- libcurl $\geq 7.21.0$
- libltdl ≥ 2.2
- sqlite || mysql || postgres



Agenda

- GUNet 0.9.x Release Status
- GUNet 0.9.x Features
- GUNet 0.9.x System Overview
- **GUNet 0.9.x APIs**



APIs: Function Pointers

- C has first-class, higher-order functions
- GNUnet uses those



```
void qsort(void *base, size_t nmemb, size_t size,  
           int(*compar)(const void *, const void *));
```

```
static int
```

```
cmpstringp(const void *p1, const void *p2) {  
    return strcmp(* (char * const *) p1,  
                 * (char * const *) p2);    }
```

```
int main(int argc, char *argv[]) {  
    qsort(&argv[1], argc - 1, sizeof(argv[1]),  
         &cmpstringp); }
```



APIs: Inner Functions

- C has first-class, higher-order functions
- GNUnet uses those
- GNU GCC has inner functions
- GNUnet does **not** use inner functions



APIs: Function Pointers and Closures

- C has first-class, higher-order functions
- GUNet uses those
- GNU GCC has inner functions
- GUNet does **not** use inner functions
- GUNet passes a `void * closure (cls)` as an explicit first argument to all higher-order functions



Example: GNUnet's directory scan API

```
typedef int (*GNUNET_FileNameCallback)  
  (void *cls , const char *filename );
```

```
int  
GNUNET_DISK_directory_scan (const char *dirName ,  
  GNUNET_FileNameCallback callback ,  
  void *callback_cls );
```



- Header includes many other headers
- Should be included after `platform.h`
- Provides OS independence / portability layer
- Provides higher-level IPC API (message-based)
- Provides some data structures (Bloom filter, hash map, heap, doubly-linked list)
- Provides configuration parsing
- Provides cryptographic primitives (AES-256, SHA-512, RSA, (P)RNG)
- Use: `GNUNET_malloc`, `GNUNET_free`, `GNUNET_strdup`, `GNUNET_snprintf`, `GNUNET_asprintf`, `GNUNET_log`, `GNUNET_assert`



APIs: `GNUNET_assert` and `GNUNET_break`

- `GNUNET_assert` aborts execution if the condition is false (0); use when internal invariants are seriously broken and continued execution is unsafe
- `GNUNET_break` logs an error message if the condition is false and then continues execution; use if you are certain that the error can be managed and if this has to be a programming error with the local peer
- `GNUNET_break_op` behaves just like `GNUNET_break` except that the error message blames it on other peers; use when checking that other peers are well-behaved
- `GNUNET_log` should be used where a specific message to the user is appropriate (not for logic bugs!); `GNUNET_log_strerror` and `GNUNET_log_strerror_file` should be used if the error message concerns a system call and `errno`



- Part of `libgnunetutil`
- Main event loop
- Each *task* is supposed to never block (disk IO is considered OK)
- SCHEDULER can be used to schedule tasks based on IO being ready, timeouts or completion of other tasks
- Each task has a unique 64-bit `GNUNET_SCHEDULER_TaskIdentifier` that can be used to *cancel* it
- The event loop is typically started using the higher-level `PROGRAM` or `SERVICE` abstractions



The scheduler provides a somewhat tricky way to install a function that will be run on shutdown:

```
static void
my_shutdown (void *cls ,
             const struct GNUNET_SCHEDULER_TaskContext *tc)
{
    GNUNET_assert (0 != (tc->reason & GNUNET_SCHEDULER_REASON_SHUTDOWN));
    GNUNET_CORE_disconnect (core);
}

static void
my_run (...)
{
    GNUNET_SCHEDULER_add_delayed (GNUNET_TIME_UNIT_FOREVER_REL,
                                  &my_shutdown, NULL);
}
}
```



- Part of `libgnunetutil`
- Used to receive requests from service APIs
- For example, GET/PUT requests from DHT API
- Main uses: register handler, transmit response to client



- Used to define message types
- Each message in GUNet begins with 4 bytes: type & size
- 64k message types, up to 64k of data per message
- You will need to define some message type(s) for your services



The STATISTICS service provides an easy way to track performance information:

```
struct GNUNET_STATISTICS_Handle *
GNUNET_STATISTICS_create (const char *subsystem,
                          const struct GNUNET_CONFIGURATION_Handle *cfg);

void
GNUNET_STATISTICS_set (struct GNUNET_STATISTICS_Handle *handle,
                      const char *name,
                      uint64_t value, int make_persistent);

void
GNUNET_STATISTICS_update (struct GNUNET_STATISTICS_Handle *handle,
                          const char *name,
                          int64_t delta, int make_persistent);
```

With this, you can then use `gnunet-statistics` to inspect the current value of the respective statistic.



The TESTING library provides an easy way to setup testbeds:

```
struct GNUNET_TESTING_PeerGroup *  
GNUNET_TESTING_peergroup_start (const struct GNUNET_CONFIGURATION_Handle *cfg ,  
                                unsigned int total ,  
                                struct GNUNET_TIME_Relative timeout ,  
                                GNUNET_TESTING_NotifyConnection connect_cb ,  
                                GNUNET_TESTING_NotifyCompletion peergroup_cb ,  
                                void *peergroup_cls ,  
                                const struct GNUNET_TESTING_Host *hostnames);
```



Other GNUnet Directories in SVN

- `svn/GNUnet` — is GNUnet 0.8.x (do NOT use this!)
- **`svn/gnunet`** — is GNUnet 0.9.x
- `svn/gnunet-java` — Java bindings for GNUnet 0.9.x
- **`svn/gnunet-ext`** — template for writing C extensions to GNUnet
- `svn/gnunet-java-ext` — template for writing Java extensions to GNUnet
- `svn/gnunet-gtk` — Gtk GUIs (including `gnunet-setup`)
- `svn/gnunet-cocoa,fuse,qt,planetlab,qt,update` — experimental, defunct or legacy (ignore!)



Using gnunet-ext

- First figure out the build system and how to compile the existing code!
- Do change “ext” (extension) to a project-specific name everywhere
- `src/template/` in `svn/gnunet/` might also be worth a look
- Do update AUTHORS, README, etc.
- Do consider adding man pages
- Do install configuration defaults to `share/gnunet/config.d/`
- Do define your own protocol numbers (`gnunet_protocols_ext.h`)
- Feel free to add additional directories (“ext” is just a starting point)