

A Generic Mailbox via XMPP

Chris Pickel <chpickel@stwing.upenn.edu>

Library example

```
void stream_auth(wockd::stream* s) {
    s->feature("http://jabber.org/features/iq-auth");

    s->iqs().add_rule(sigc::ptr_fun(get_iq_auth))
        .name("jabber:iq:auth", "query")
        .type(wockd::iq::get);
    s->iqs().add_rule(sigc::ptr_fun(set_iq_auth))
        .name("jabber:iq:auth", "query")
        .type(wockd::iq::set);
}

void get_iq_auth(wockd::iq::incoming& i) {
    wockd::node(iq_auth, "query");
    n.insert_tag("username");
    n.insert_tag("password");
    n.insert_tag("resource");
    i.yield_result(&n);
}

void set_iq_auth(wockd::iq::incoming& i) {
    if (/* data valid */) {
        /* bind stream */
        i.yield_result();
    } else {
        i.yield_error(wockd::error::modify,
            wockd::error::not_acceptable);
    }
}

void stream_torrent(wockd::stream* s) {
    s->iqs().add_rule(sigc::ptr_fun(add_download))
        .name("http://wockd.org/ns/torrent", "add")
        .type(wockd::iq::set);
}

void add_download(wockd::iq::incoming& i) {
    new add_download_responder(i);
}

struct add_download_responder {
    wockd::iq::incoming i;
    add_download_responder(wockd::iq::incoming& i) {
        /* start HTTP request */
    }
    void http_done() {
        /* start download */
        i.yield_result();
    }
    void http_failed() {
        i.yield_error(/* ... */);
    }
}

wockd::socket* initialize_stream(int fd) {
    wockd::stream* s = new wockd::stream(fd);
    stream_auth(s);
    stream_torrent(s);
}
```

Abstract

More services are available over the internet, but users are not always connected, and they use the services on different devices. We want to use protocols that can deal with these circumstances.

Rather than develop new protocols that no one will use, the goal was to provide an extensible and expressive system to wrap existing services. This is done in XMPP, an extensible messaging protocol.

Results

The product of this project is wockd, a basic XMPP server written in C++. It compiles and runs on both Linux and Mac OS X; it runs over IPv4 and IPv6, and includes the foundations for running a BitTorrent client over XMPP.

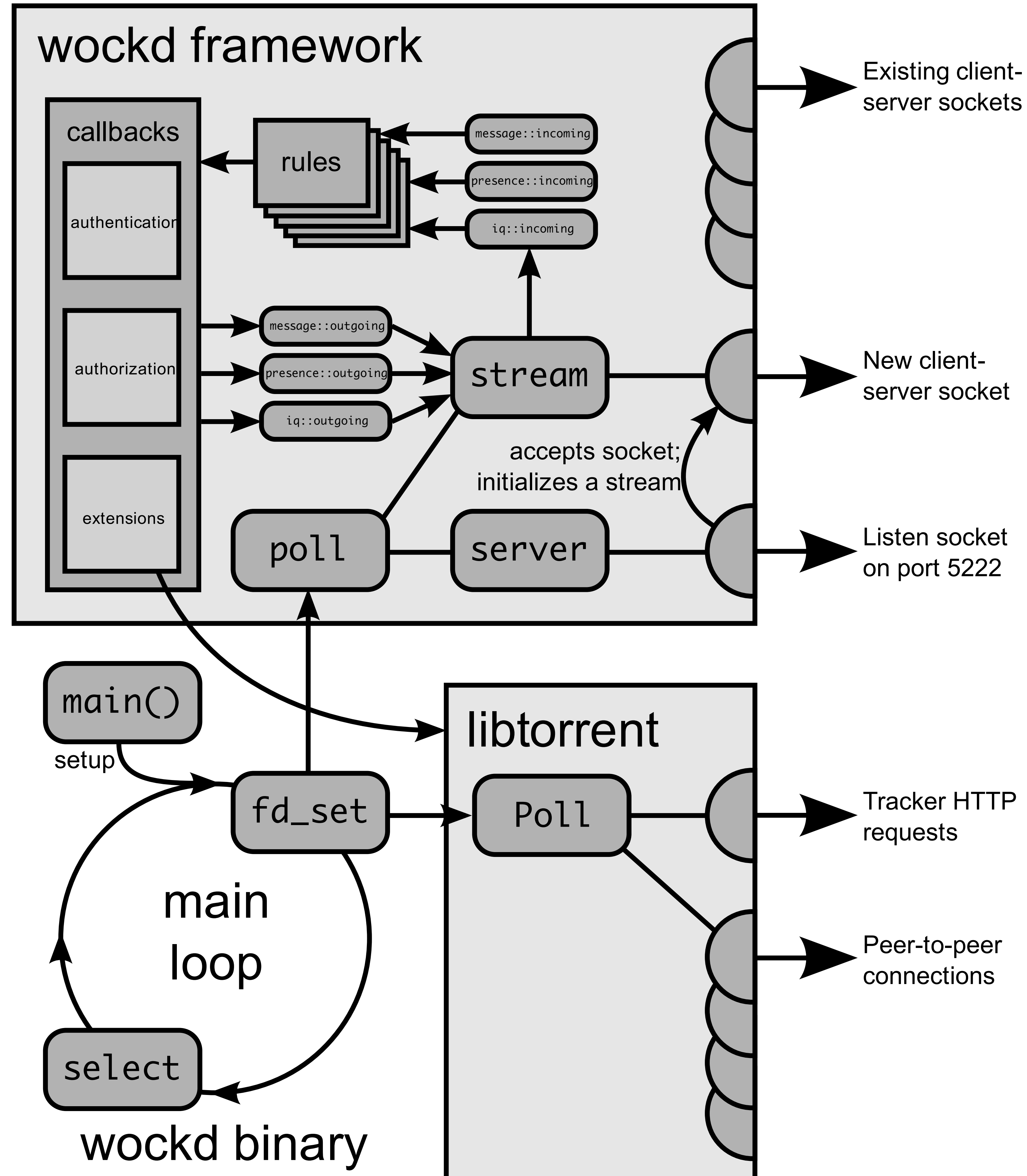
In order to build wockd, it was necessary to design the server such that its main event loop could be integrated with another library. In addition to integration with libtorrent, there is an experimental poll class that runs within the glib event loop.

Future

There's still work to be done to get a fully-functional BitTorrent client to run within wockd; there are also other application domains that could benefit from being run remotely.

In particular, I would like to use the glib-based poll class to wrap libpurple, and provide an IM transport to various instant messaging services.

wockd framework



Protocol example

```
C: <stream:stream
  xmlns="jabber:client"
  xmlns:stream="http://etherx.jabber.org/streams"
  to="jabber.wockd.org">
S: <stream:stream
  xmlns="jabber:client"
  xmlns:stream="http://etherx.jabber.org/streams"
  from="jabber.wockd.org">
S: <stream:features
  <auth xmlns="http://jabber.org/features/iq-auth"/>
</stream:features>
C: <iq id="0" type="get">
  <query xmlns="jabber:iq:auth"/>
</iq>
S: <iq id="0" type="result">
  <query xmlns="jabber:iq:auth">
    <username/>
    <password/>
    <resource/>
  </query>
</iq>
C: <iq id="1" type="set">
  <query xmlns="jabber:iq:auth">
    <username>bander</username>
    <password>****</password>
    <resource>snatch</resource>
  </query>
</iq>
S: <iq id="1" type="error">
  <error type="modify">
    <not-acceptable
      xmlns="urn:ietf:params:xml:ns:xmpp-stanzas"/>
  </error>
</iq>
C: <iq id="2" type="set">
  <query xmlns="jabber:iq:auth">
    <username>bander</username>
    <password>*****</password>
    <resource>snatch</resource>
  </query>
</iq>
S: <iq id="2" type="result"/>
C: <iq id="3" type="set">
  <add xmlns="http://wockd.org/ns/torrent">
    http://torrent.ubuntu.com:6969/file?info_hash=...
  </add>
</iq>
S: <iq id="3" type="result"/>
C: </stream:stream>
S: </stream:stream>
```