TLS Support

Getting Started

Building

To build with TLS support you'll need OpenSSL development libraries (e.g. libssl-dev on Debian/Ubuntu).

Run `make BUILD_TLS=yes`.

Tests

To run Redis test suite with TLS, you'll need TLS support for TCL (i.e. `tcl-tls` package on Debian/Ubuntu).

1. Run `./utils/gen-test-certs.sh` to generate a root CA and a server certificate.
2. Run `./runtest --tls` or `./runtest-cluster --tls` to run Redis and Redis Cluster tests in TLS mode.

Running manually

To manually run a Redis server with TLS mode (assuming `gen-test-certs.sh` was invoked so sample certificates/keys are available):

```
./src/redis-server --tls-port 6379 --port 0 \
    --tls-cert-file ./tests/tls/redis.crt \
    --tls-key-file ./tests/tls/redis.key \
    --tls-ca-cert-file ./tests/tls/ca.crt
```

To connect to this Redis server with `redis-cli`:

```
./src/redis-cli --tls \
    --cert ./tests/tls/redis.crt \
    --key ./tests/tls/redis.key \
    --cacert ./tests/tls/ca.crt
```

This will disable TCP and enable TLS on port 6379. It's also possible to have both TCP and TLS available, but you'll need to assign different ports.

To make a Replica connect to the master using TLS, use `--tls-replication yes`, and to make Redis Cluster use TLS across nodes use `--tls-cluster yes`.

Connections

All socket operations now go through a connection abstraction layer that hides I/O and read/write event handling from the caller.

Multi-threading I/O is not currently supported for TLS, as a TLS connection needs to do its own manipulation of AE events which is not thread safe. The solution is probably to manage independent AE loops for I/O threads and longer term association of connections with threads. This may potentially improve overall performance as well.

Sync IO for TLS is currently implemented in a hackish way, i.e. making the socket blocking and configuring socket-level timeout. This means the timeout value may not be so accurate, and there would be a lot of syscall overhead.

However I believe that getting rid of syncio completely in favor of pure async work is probably a better move than trying to fix that. For replication it would probably not be so hard. For cluster keys migration it might be more difficult, but there are probably other good reasons to improve that part anyway.

To-Do List

- [ ] redis-benchmark support. The current implementation is a mix of using hiredis for parsing and basic networking (establishing connections), but directly manipulating sockets for most actions. This will need to be cleaned up for proper TLS support. The best approach is probably to migrate to hiredis async mode.
- [ ] redis-cli `--slave` and `--rdb` support.

Multi-port

Consider the implications of allowing TLS to be configured on a separate port, making Redis listening on multiple ports:

1. Startup banner port notification
2. Proctitle
3. How slaves announce themselves
4. Cluster bus port calculation