qmtunnel Documentation

Release 0.1

Nikolay N. Karikh

Sep 27, 2017

Contents:

1 Introduction **3**
- 1.1 How it works . . . 3
- 1.2 Architecture . . . 5
- 1.3 License . . . 5
- 1.4 Support . . . 10
- 1.5 Copyrights . . . 10

2 Getting started **11**
- 2.1 Installation . . . 11
- 2.2 Setting up qmTunnel server instance . . . 11
- 2.3 Setting up qmTunnel GUI . . . 16
- 2.4 Connecting to qmTunnel server from GUI . . . 18
- 2.5 Creating tunnels from GUI . . . 21

3 Tunnel setup **27**
- 3.1 Tunnel settings . . . 27
- 3.2 Tunnel servers connection settings . . . 29

4 Troubleshooting **33**
- 4.1 Debug logs . . . 33
- 4.2 Draw your schema . . . 33
- 4.3 Include you configuration . . . 34

5 Building from source **37**
- 5.1 RedHat/CentOS (7) . . . 37
- 5.2 Ubuntu (14.04 LTS) . . . 38
- 5.3 Windows XP and later (32 bit) . . . 38
- 5.4 NEED HELP? . . . 39

6 Indices and tables **41**

Note: Please keep in mind that both qmTunnel and documentation are under development now. You can only use it on your own risk.

If you wish to use it in production environment, consider using our commercial support service to ensure better experience and support the project financially.

CHAPTER 1

Introduction

qmTunnel is a free cross-platform open source tunneling software allowing you to wrap up and tunnel all types of TCP, UDP or named pipe connections through a set of tunnel software servers.

You may find qmTunnel useful if you need (all features are optional):

- to secure your connection with SSL/TLS;
- to connect to hosts/networks behind NAT/firewall;
- to compress your traffic;
- to detect silent packet drops and disconnections (by enabling heartbeats);
- to allow short-time disconnections between tunnel hosts with no application disconnections;
- to add additional authentication level to tunnel hosts;
- to automatically re-establish the tunnel on disconnections (permanent tunnel);
- to establish tunnel only when needed (on demand).

Basically your application client connects to qmTunnel server instead of connecting directly to application server. Then qmTunnel server makes further connections to next qmTunner server and the last qmTunnel server in chain connects to your application server, transparently (for application client and server) transferring all application data from application client to the application server (and vice versa) and allowing to secure and tune the connections between qmTunnel servers.

How it works

The most simple case can be illustrated by the following figure:

When tunnel is configured on «host A», the connection is established in the following order:

1. Application client connects to local (on the same node or network) qmTunnel server on specified TCP or UDP port «X» on «host A».

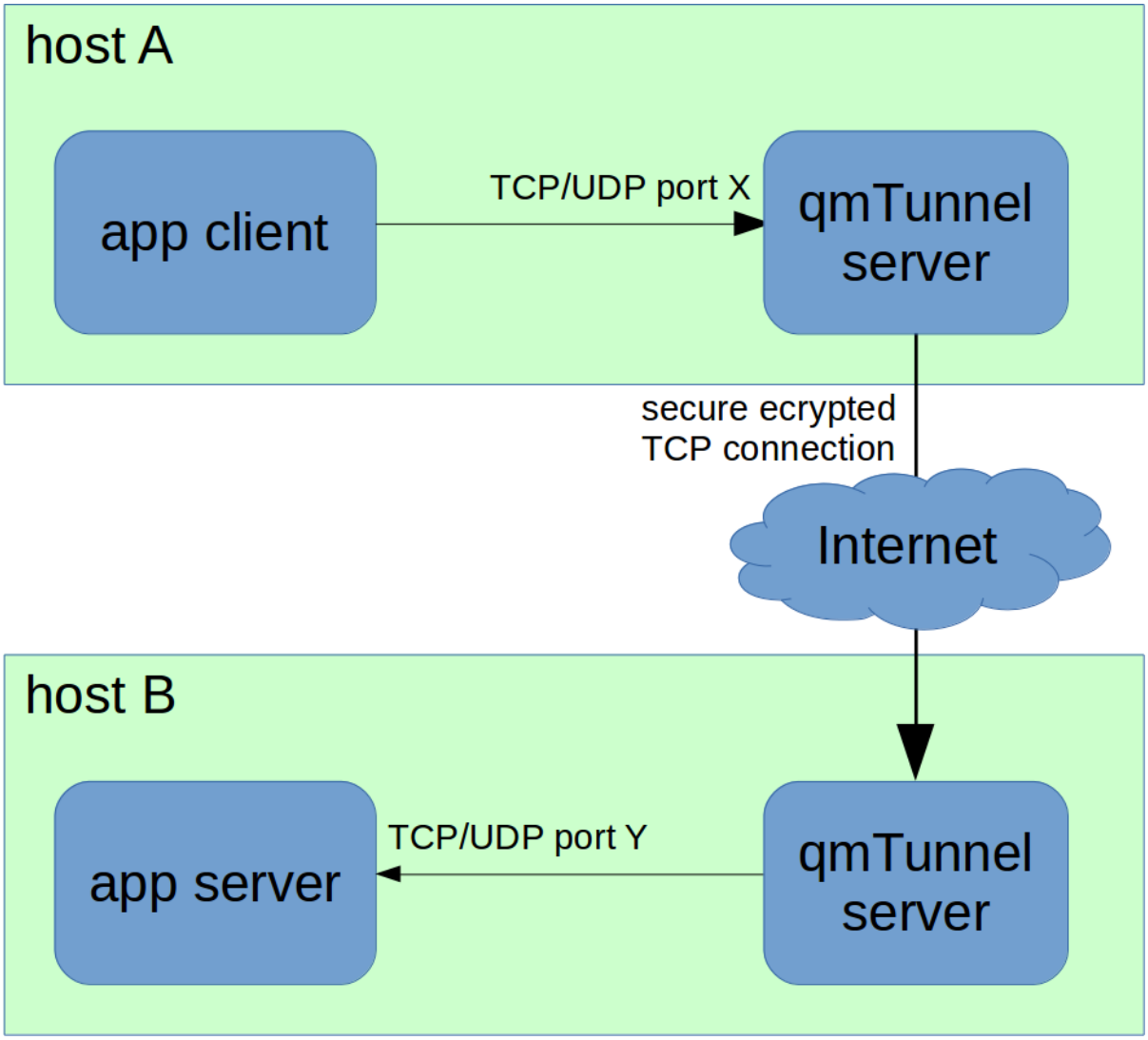


Fig. 1.1: A simple tunnel through two hosts

2. qmTunnel server on «host A» connects to qmTunnel server on «host B» and establishes encrypted (SSL/TLS) connection.
3. qmTunnel server on «host B» connects to local (same node or network) application server on specified TCP or UDP port «Y». *«X» might be the same as «Y», or not.*
4. All the data transmitted by the application client will be delivered through the tunnel to the application server and vice versa.

Application might be any application or service using TCP, UDP or Unix sockets. For example:

- E-mail (SMTP, POP3, IMAP)
- Databases or data storages (MySQL, PostgreSQL, Oracle, SQL Server, etc.)
- Remote desktop and shell (VNC, RDP, SSH, etc.)
- DNS
- any other

In case you need to provide tunnel access to your LAN it's possible to do so:

It's also possible that host «B» is not available from the Internet and can't accept incoming connections, e.g. located behind NAT/firewall or doesn't have real IP address or domain name. In this case «remote» (or «reverse») tunnel can help:

«Remote» (or «reverse») tunnel means that it's «host B» who initiates the connection to «host A», and also the tunnel needs to be initiated on «host B» instead of «host A».

The next possible scenario is that both application client and server are behind NAT/firewall and incoming connections can't be accepted from outside. In this case you will need a third host acting as communication server/proxy:

In this case you need to create 2 separate tunnels:

1. On «host A»: local forwarding tunnel from port «X» to «host C» port «Z».
2. On «host B»: remote forwarding tunnel from «host C» port «Z» to local port «Y».

Your tunnel's host chain length is not limited if you need to pass through a few gateways:

Architecture

qmTunnel consists of 2 modules:

- **qmTunnel-server** — server module which needs to be started on all tunnel hosts (at least two). It's possible to run qmtunnel-server as GUI application or as background console application (use `-daemon` command line parameter).
- **qmTunnel-gui** — GUI which connects to qmtunnel-server instances (including remote ones) and allows to configure them and create/edit/monitor tunnels.

qmTunnel is a free open source cross-platform application and runs on Linux, Windows and possibly (haven't tested yet) MacOS.

To build and run qmTunnel, you only need Qt4/Qt5 and OpenSSL libraries.

License

qmtunnel is released under GNU General Public License 3.0, with the additional special exception to link portions of this program with the OpenSSL library. See LICENSE file for more details.

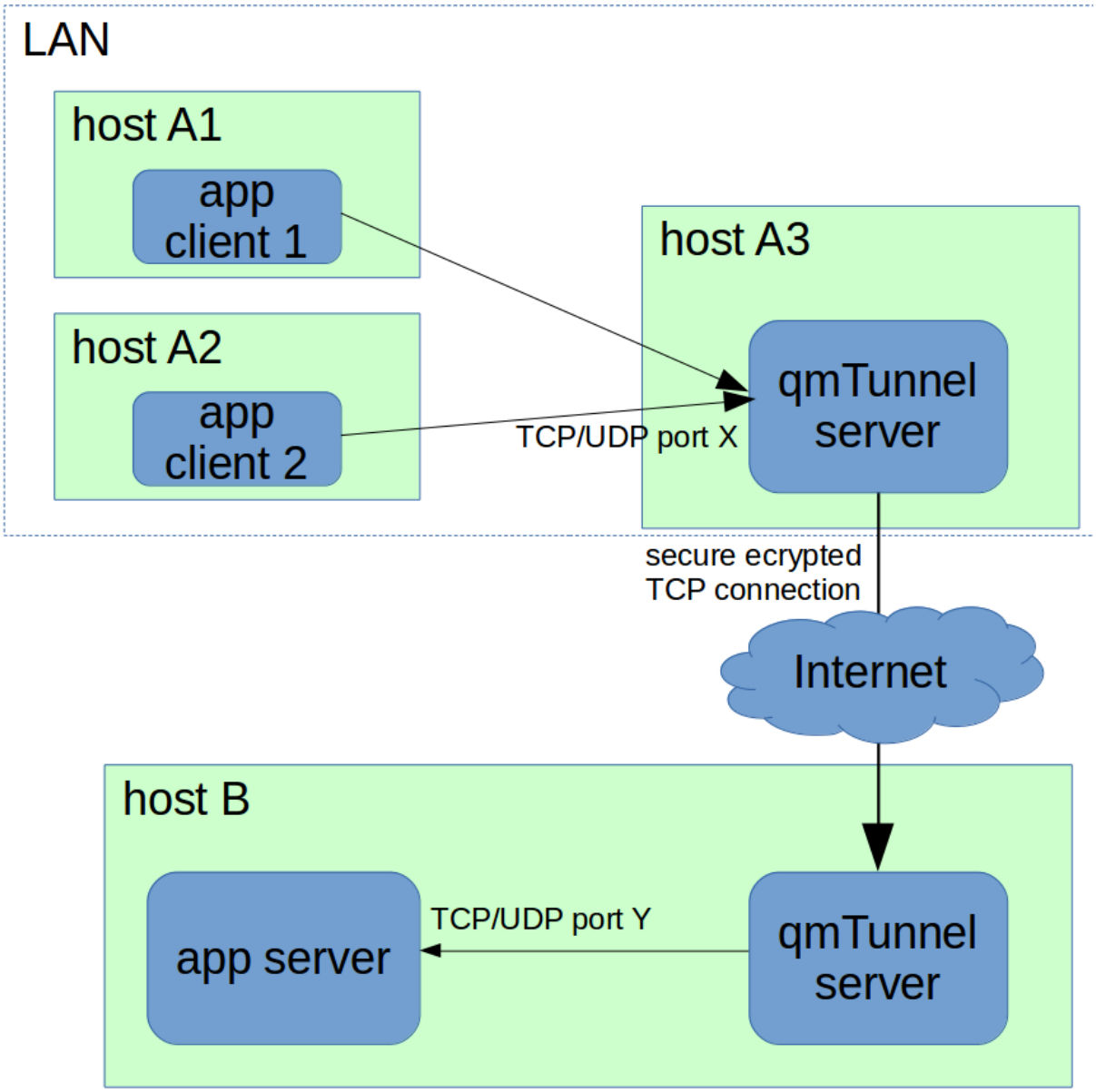


Fig. 1.2: Providing tunnel to other hosts on the same network

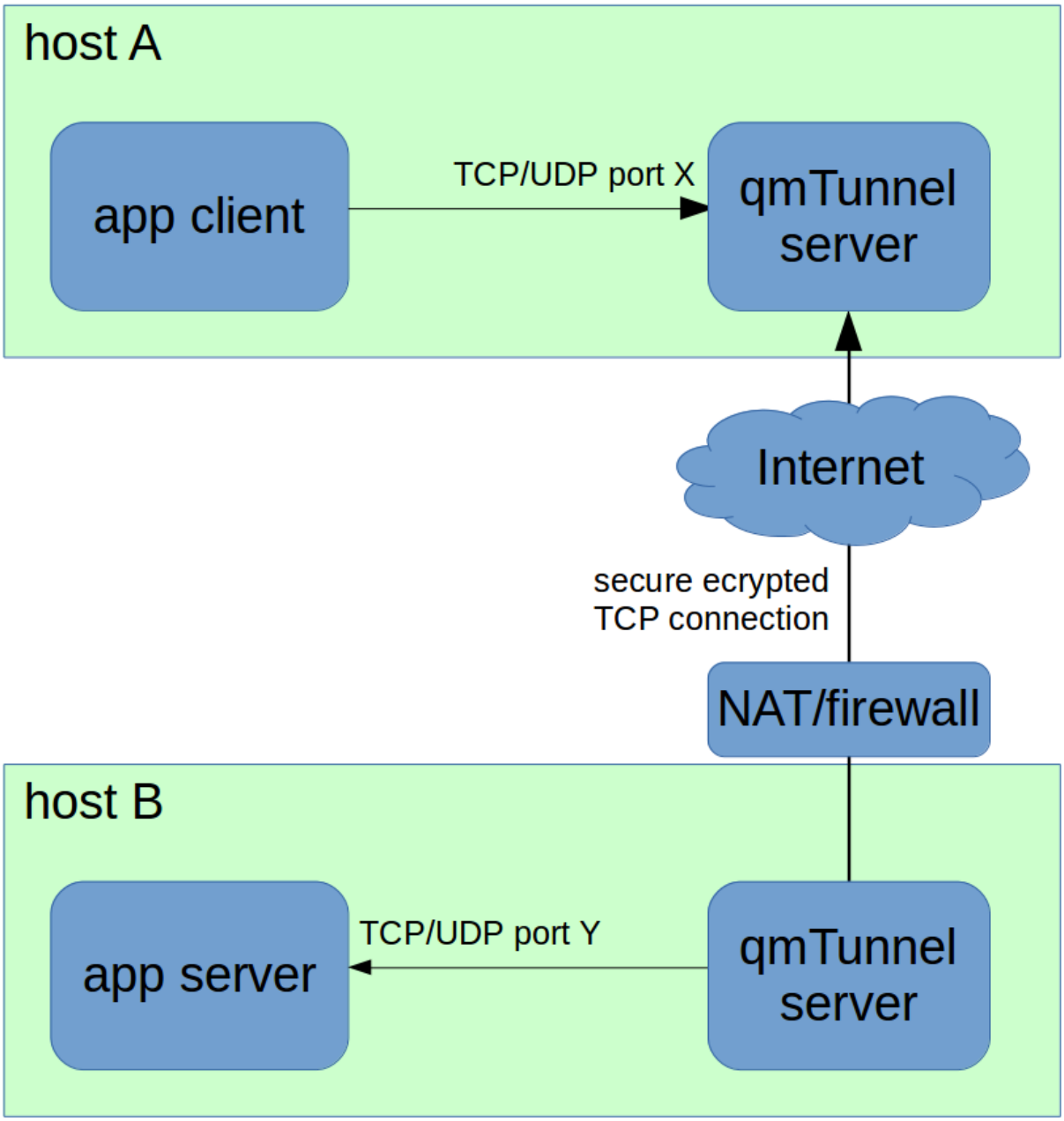


Fig. 1.3: Remote (reverse) tunnel

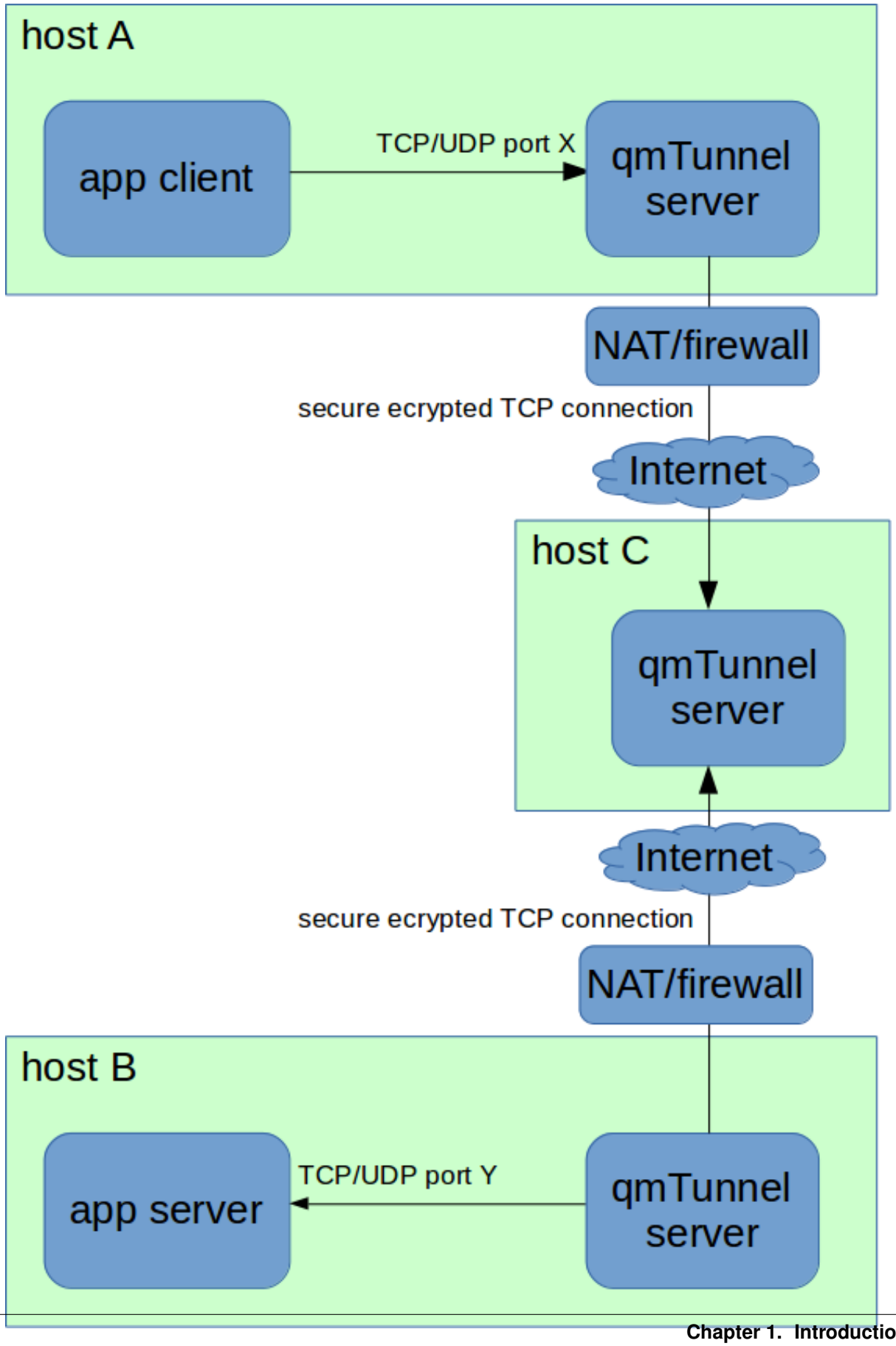


Fig. 1.4: Using extra qmTunnel server as communication server

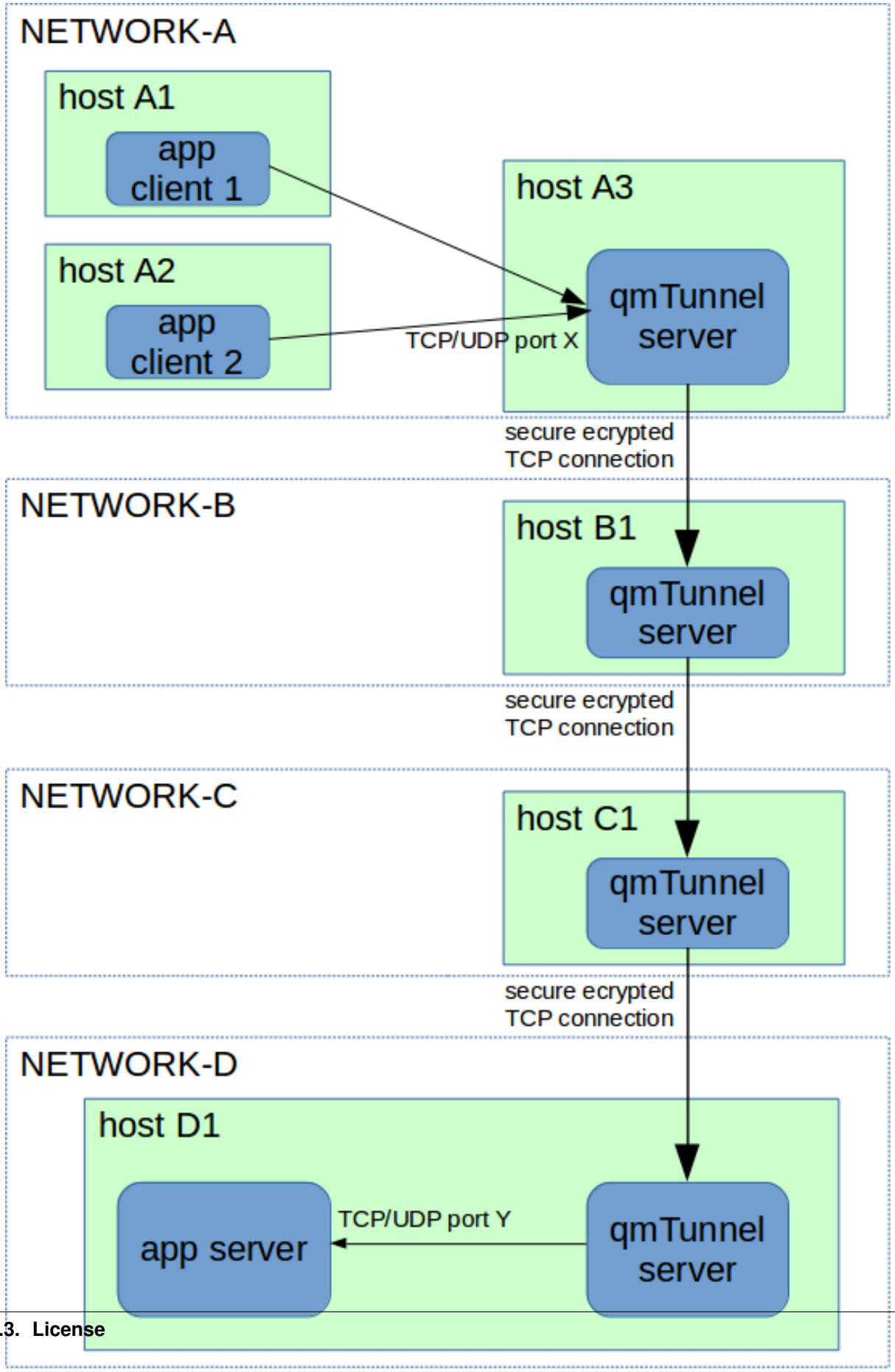


Fig. 1.5: Tunnel through several gateways

Support

qmtunnel is open-source project, which means it's considered to be supported by the community.

However if you wish to use it in production environment, commercial support is also available from the author and maintainer of this project. Contact support@qmtunnel.com for details. This way you can also support the project.

Copyrights

CHAPTER 2

Getting started

You may find qmTunnel useful if you need (all features are optional):

- to secure your connection with SSL/TLS;
- to connect to hosts/networks behind NAT/firewall;
- to compress your traffic;
- to detect silent packet drops and disconnections (by enabling heartbeats);
- to allow short-time disconnections between tunnel hosts with no application disconnections;
- to add additional authentication level to tunnel hosts;
- to automatically re-establish the tunnel on disconnections (permanent tunnel);
- to establish tunnel only when needed (on demand).

Installation

You can either download binaries or build the project from sources.

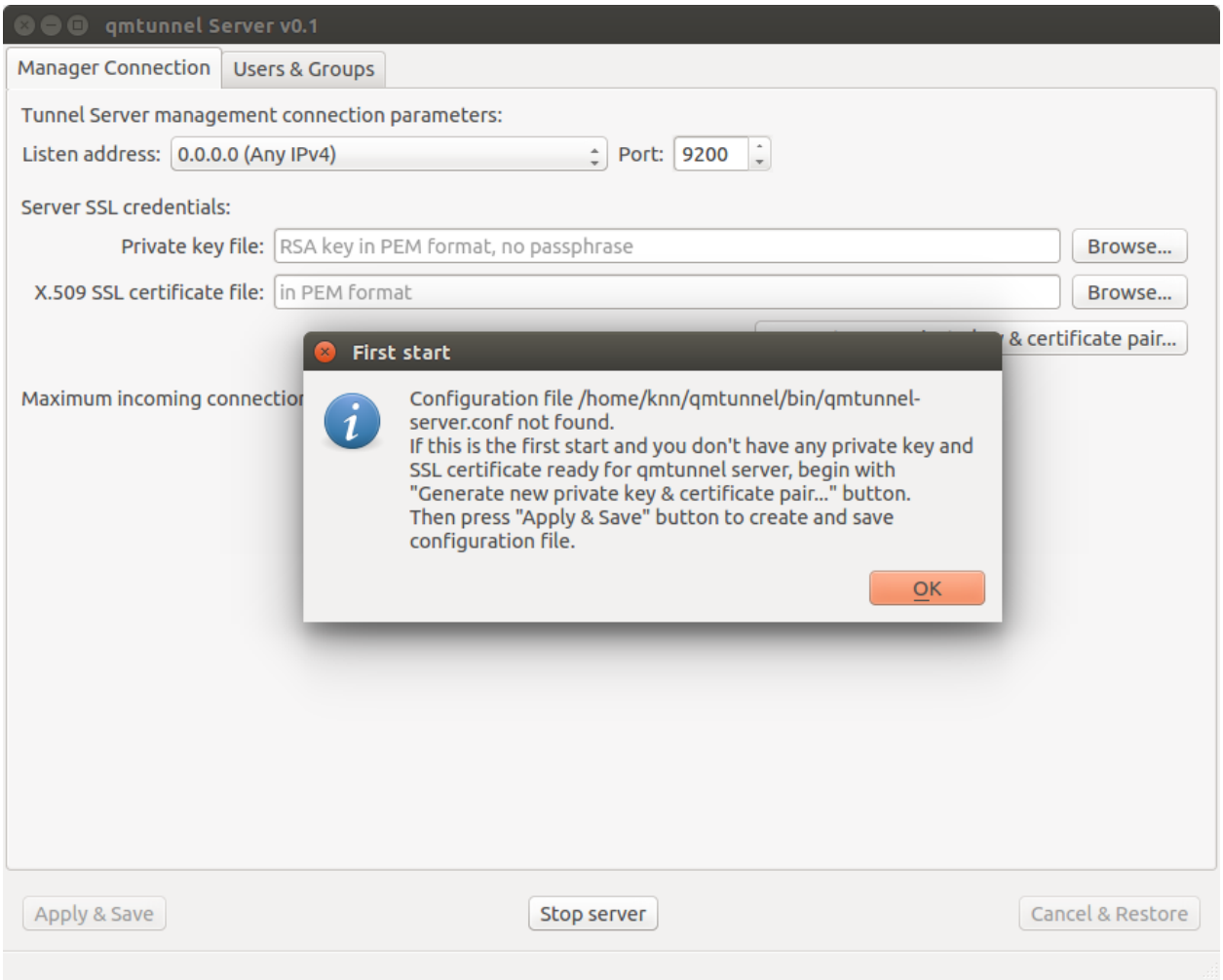
Binaries download link: http://qmtunnel.com/download/

At the moment the binaries are provided without any installation tools or scripts and for Windows (compiled in 32-bit) only. You just need to unpack the files and put them into some directory.

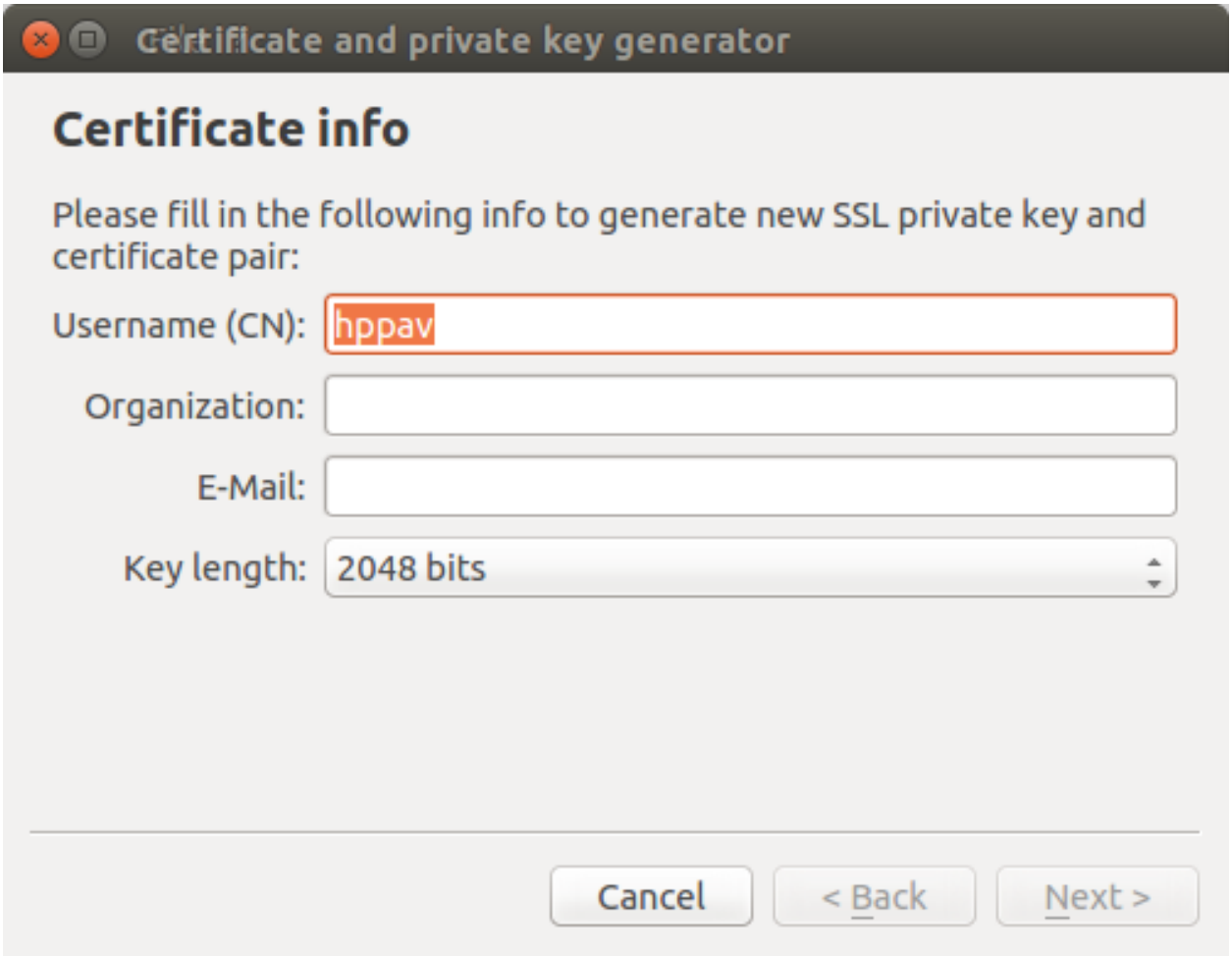
You can also see section *Building from source* for build instructions on Linux platforms.

Setting up qmTunnel server instance

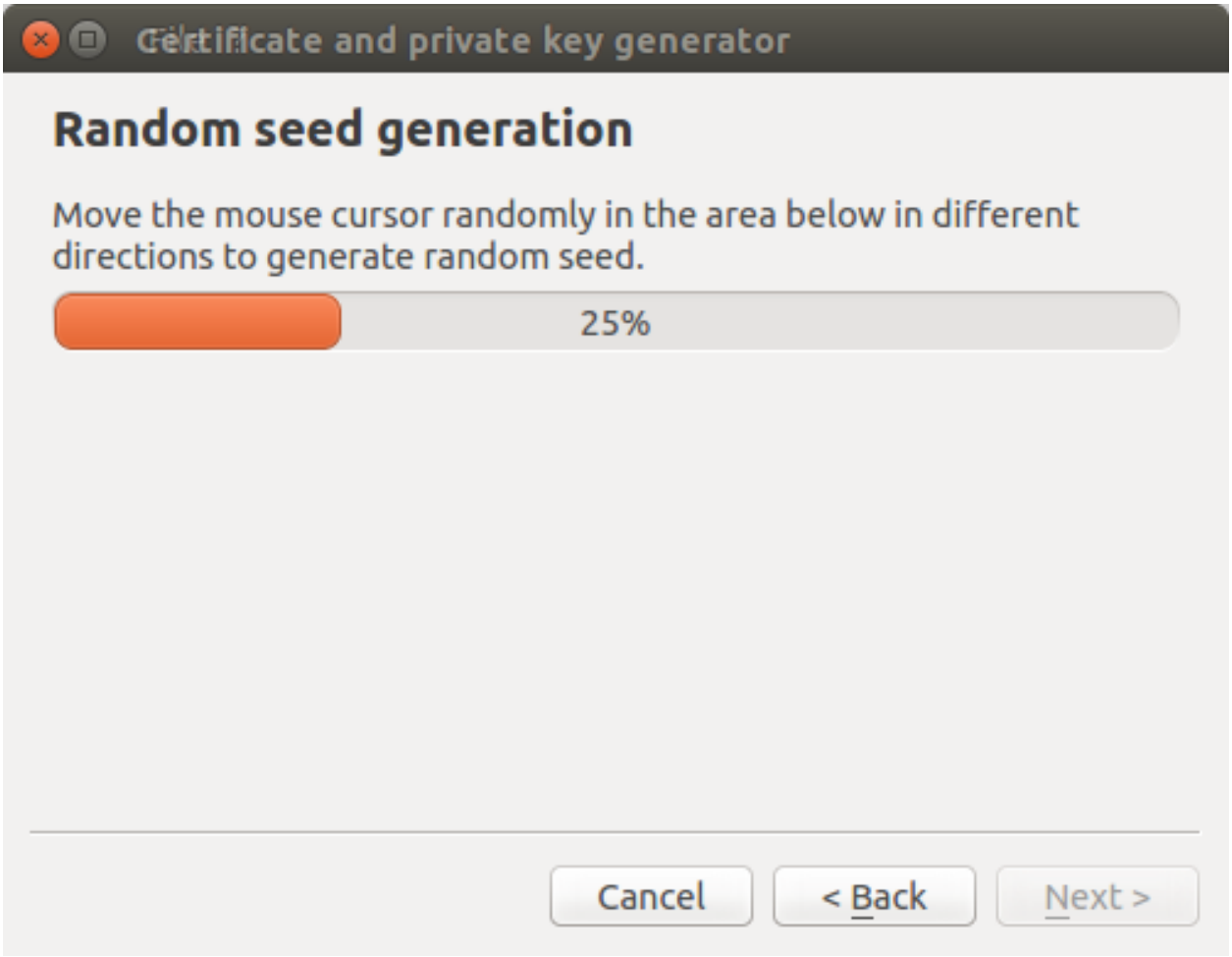
When you first run `qmtunnel-server` process in GUI mode, it will ask you to set up your qmTunnel server instance certificate:



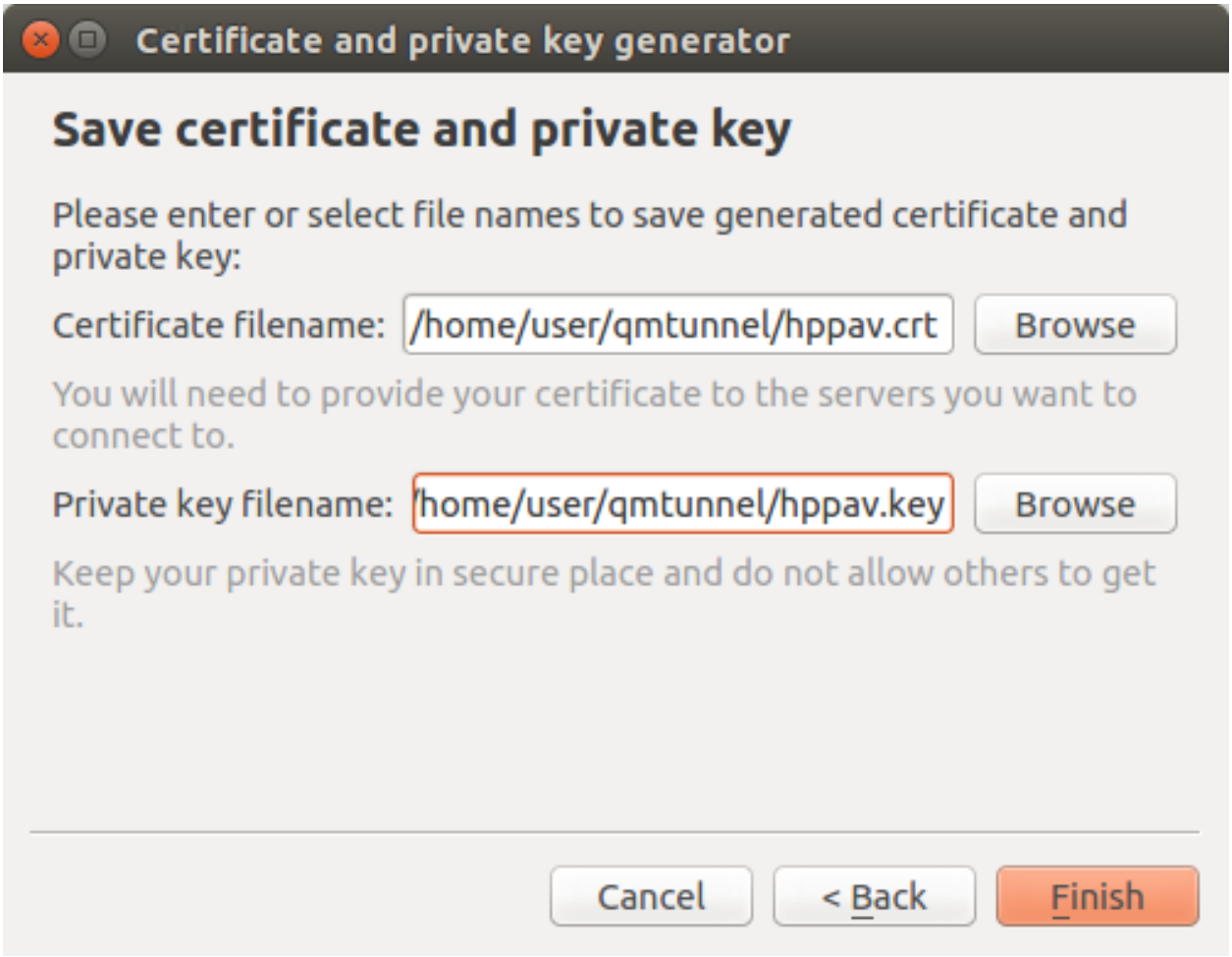
Click `OK` and then `Generate new private key & certificate pair...`.



Edit `Username (CN)` and `Key length` if neccessary and press `Next >`.



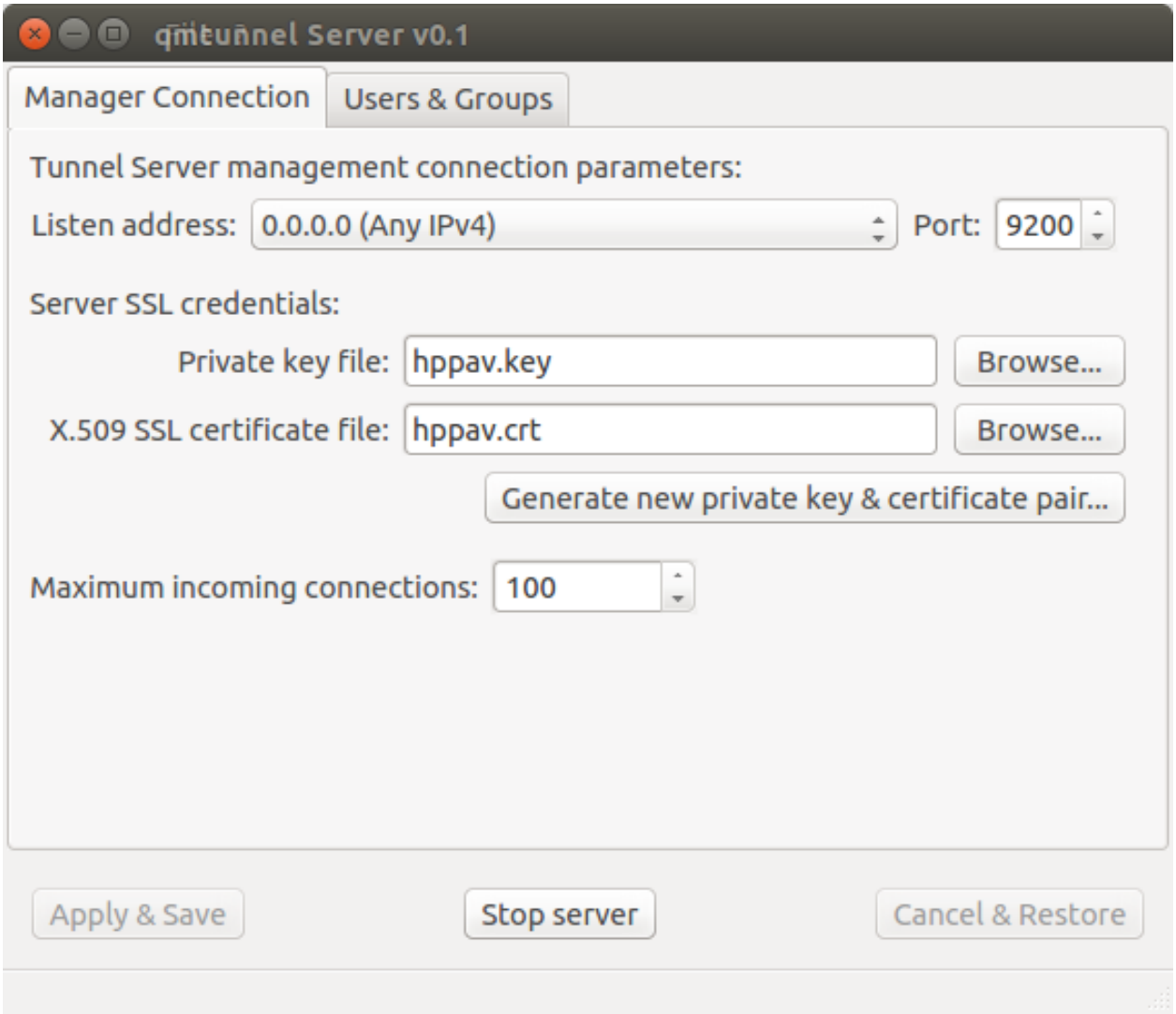
Randomly move your mouse cursor over empty space inside the wizard window to generate better random seed. Then press `Next >`.



Edit file paths and press `Finish`.

Note: Please ensure that you keep your private key in a safe place and do not give access to it to anyone except this qmTunnel server instance.

Your server private key and certificate is ready! Press `Apply & Save` button in the bottom.



You can also change the TCP `Port` on which qmTunnel operates (if neccessary) and choose specific `Listen address` if you don't want qmTunnel to accept connections from all available networks.

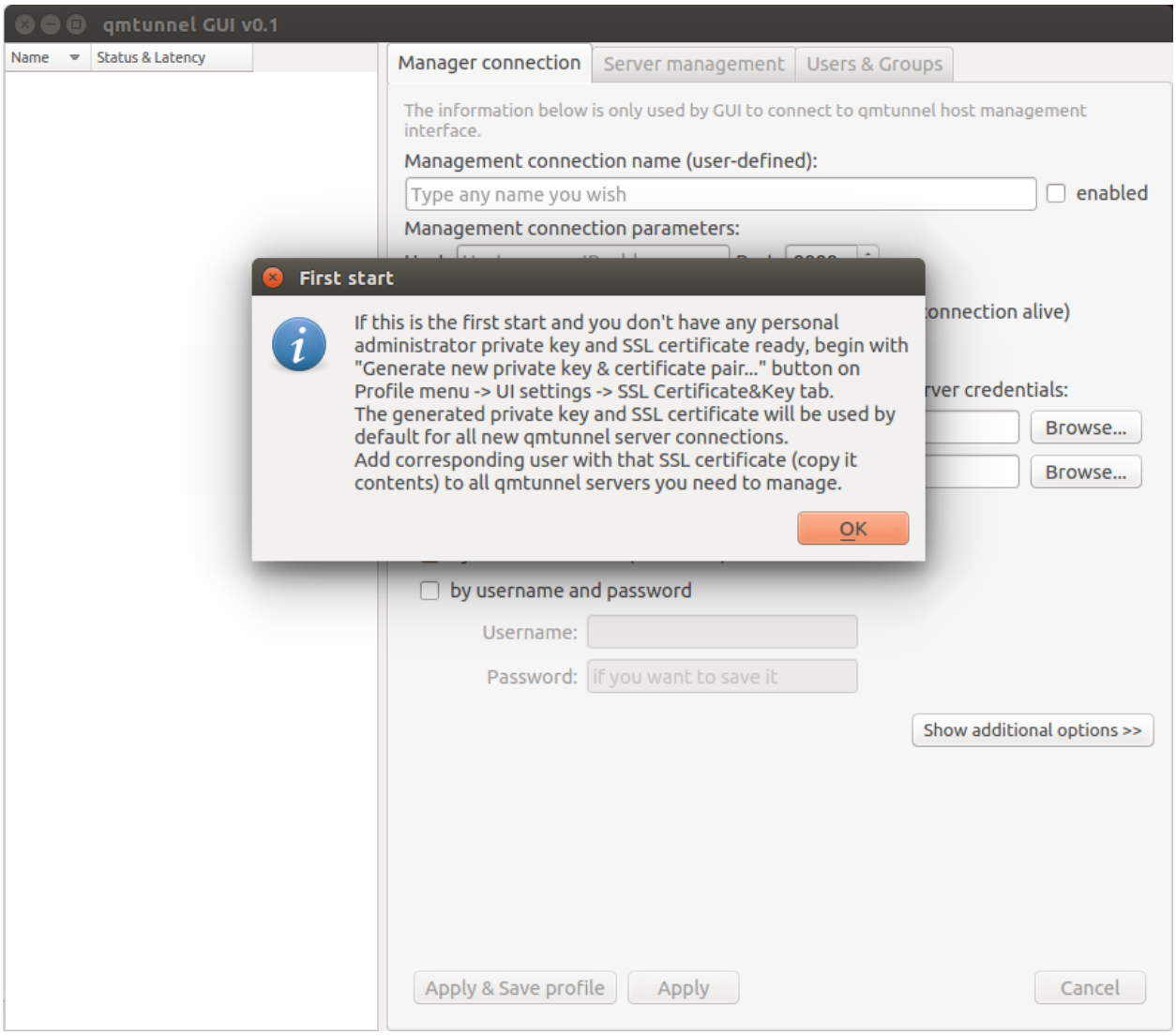
OK, now our first qmTunnel server is up & running.

Repeat the above setup steps on at least one other host to have at least two qmTunnel servers running. Let's suppose that the first qmTunnel server would be at the client side, and the second one would be on destination server or network.

Setting up qmTunnel GUI

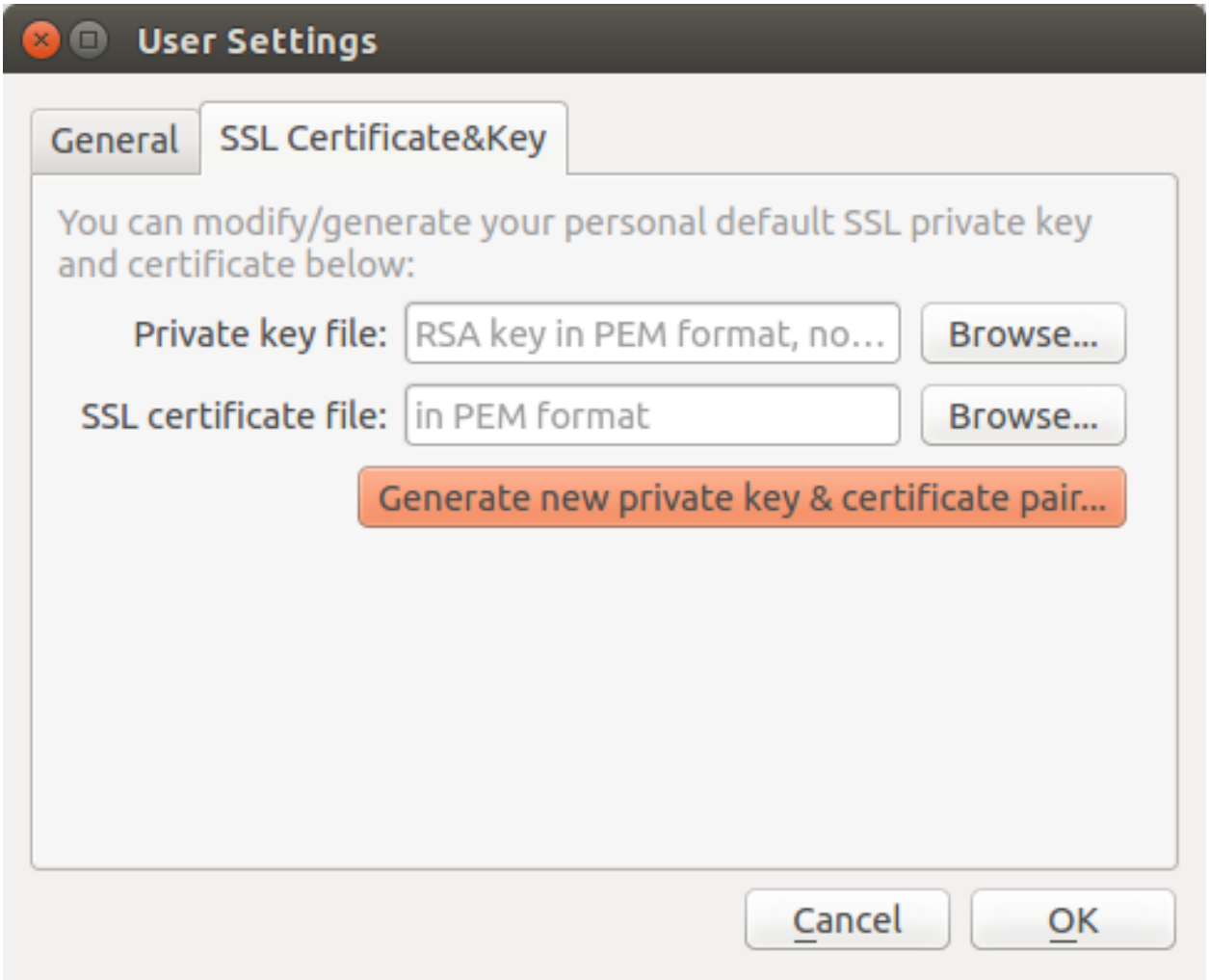
In order to create, modify and monitor tunnels and manage qmTunnel servers you need to run separate application called `qmTunnel-gui`.

When you first run `qmtunnel-gui`, it will also ask you to set up your personal private key and certificate which will be used to connect to and manage qmTunnel servers:



Click `OK` and `User Settings` dialog would pop up, where you can then press `Generate new private key & certificate pair...` button and then repeat the same steps as for server certificate.

If you missed `User Settings` dialog, you can call it from main window menu `Profile` -> `UI Settings...` and then go to `SSL Certificate&Key` tab.

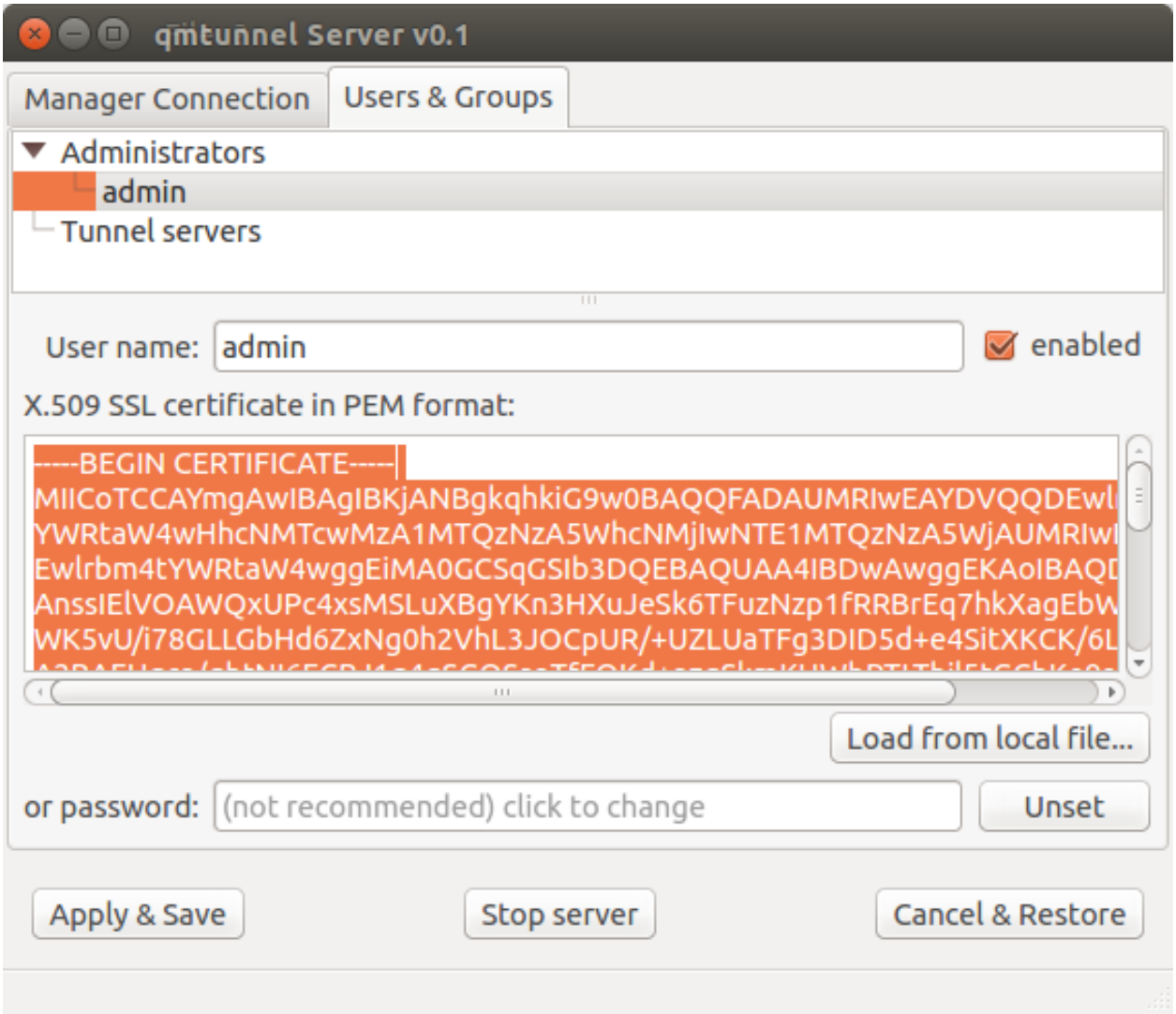


This private key and certificate will be used by default to connect to qmTunnel servers.

Connecting to qmTunnel server from GUI

qmTunnel server requires user or other server's certificate to authorize incoming connection.

Once you have created your personal certificate in `qmtunnel-gui` (see above), you can copy&paste the contents of this certificate in `qmTunnel-server`:



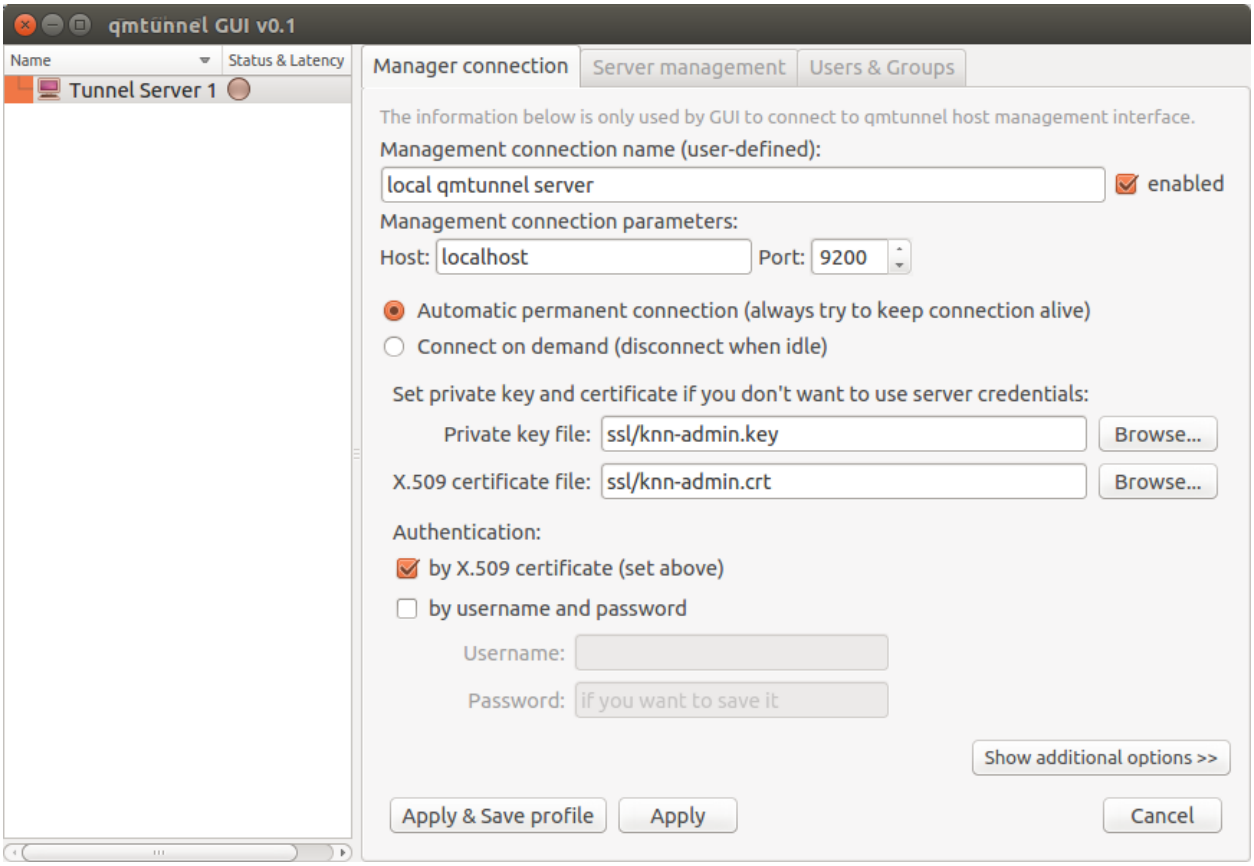
In `qmtunnel-server` user interface there is tab called `Users & Groups` which allows to manage access to this qmTunnel server instance.

By default, there is an `Administrators` group with full access and one `admin` user without certificate. Just select this `admin` user in the list and paste your personal certificate contents into corresponding field.

Do not forget to press `Apply & Save` button after any modification.

OK, now return to `qmtunnel-gui` application.

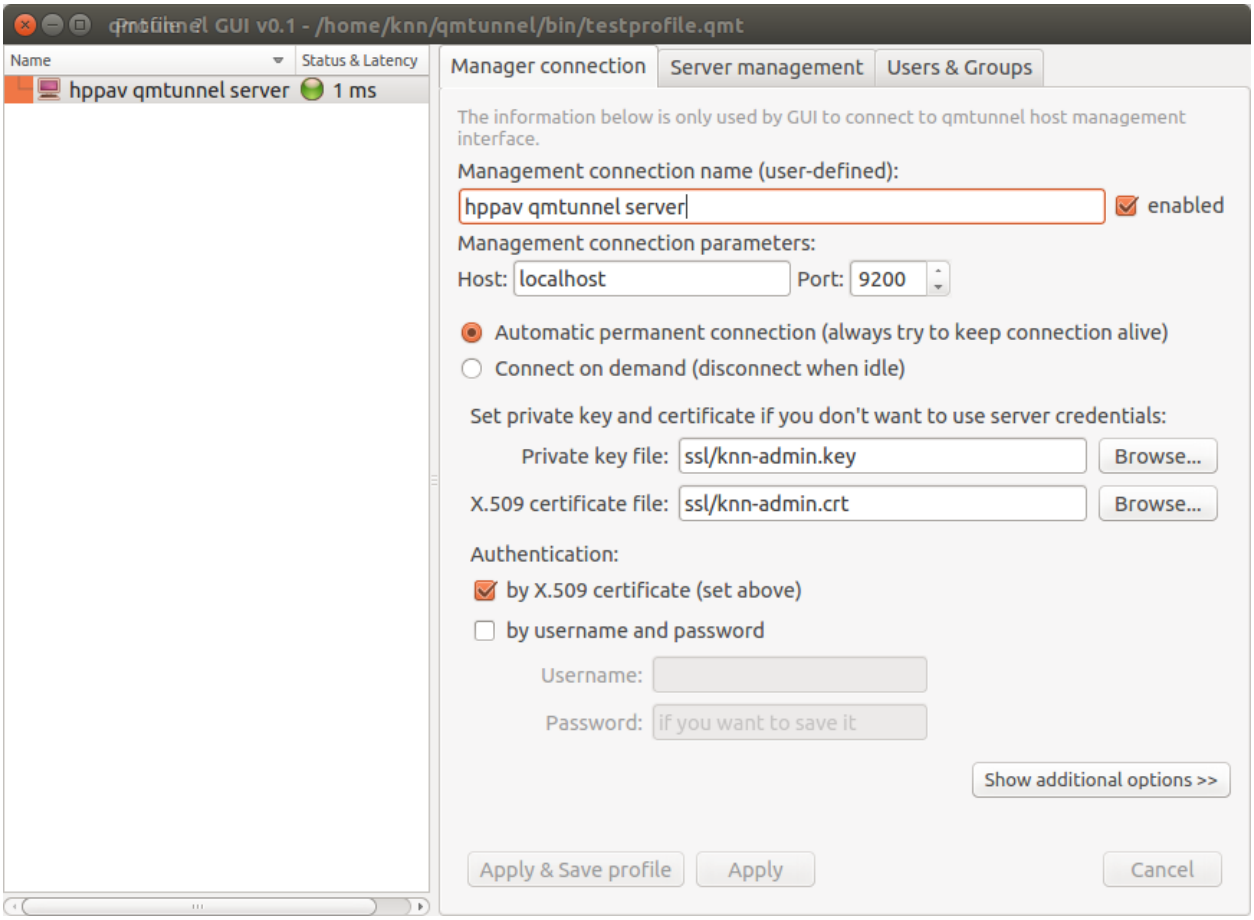
At the left part of the main window there is a list of qmTunnel servers available. At the moment it's empty, so right-click on this list and select `Add tunnel server...` from the context menu:



There is not much info required to enter here. Just enter something in the `Management connection name`, enter qmTunnel server IP-address or domain name (`localhost` if on the same host) in `Host` field, and port (if changed in `qmtunnel-server`).

There are many additional settings which are available by clicking `Show additional options` button, but in most cases you'll be fine with defaults.

OK, now press `Apply & Save profile` button and choose a file to store your profile (qmTunnel servers list) in.



Now the green indicator shows that you have successfully connected to your first qmTunnel server.

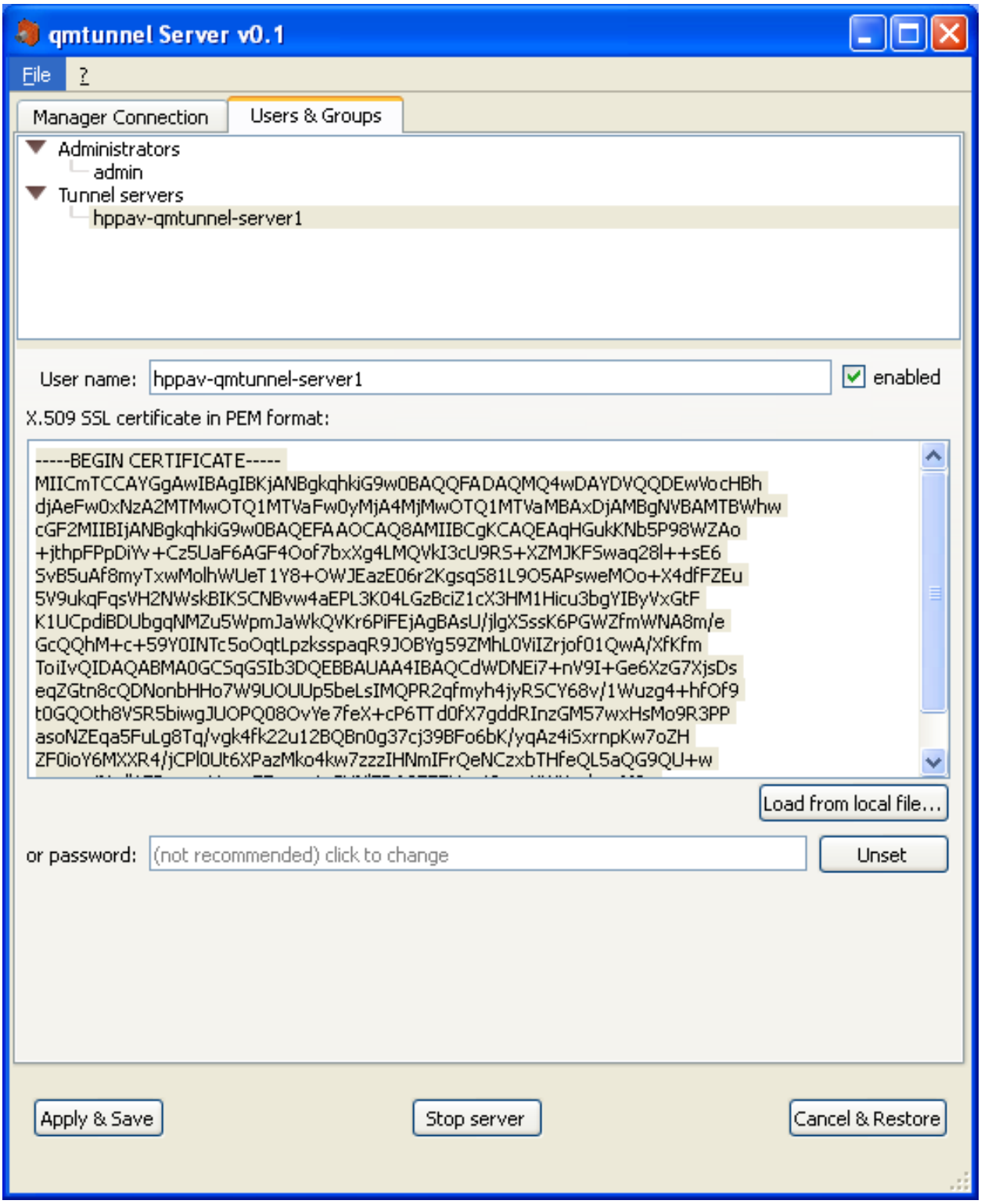
You can now use `qmtunnel-gui` to configure this qmTunnel server. Use `Server management` and `Users & Groups` tabs for this purpose.

Creating tunnels from GUI

In order to create a tunnel, you first need to ensure that any of your qmTunnel servers "know" previous qmTunnel server in chain.

In the current example, we need to authorize our first qmTunnel server on the second qmTunnel server.

There is a pre-created user group called `Tunnel servers` which we can add our first server to:



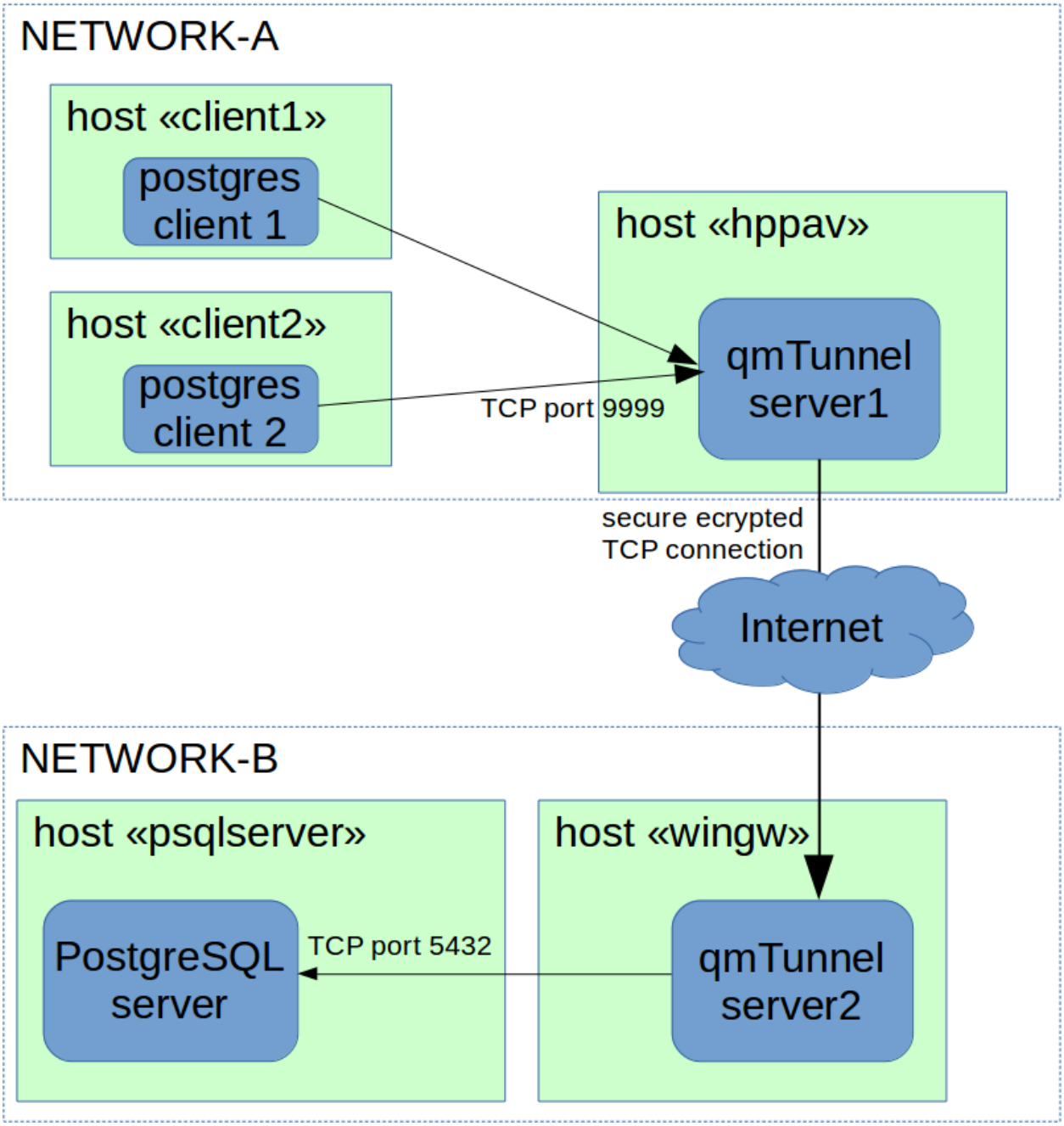
Do this on the second qmTunnel server.

To add user to the group, right-click on the group and select `Add user...`. Specify any user name your want, just make sure to paste correct server certificate.

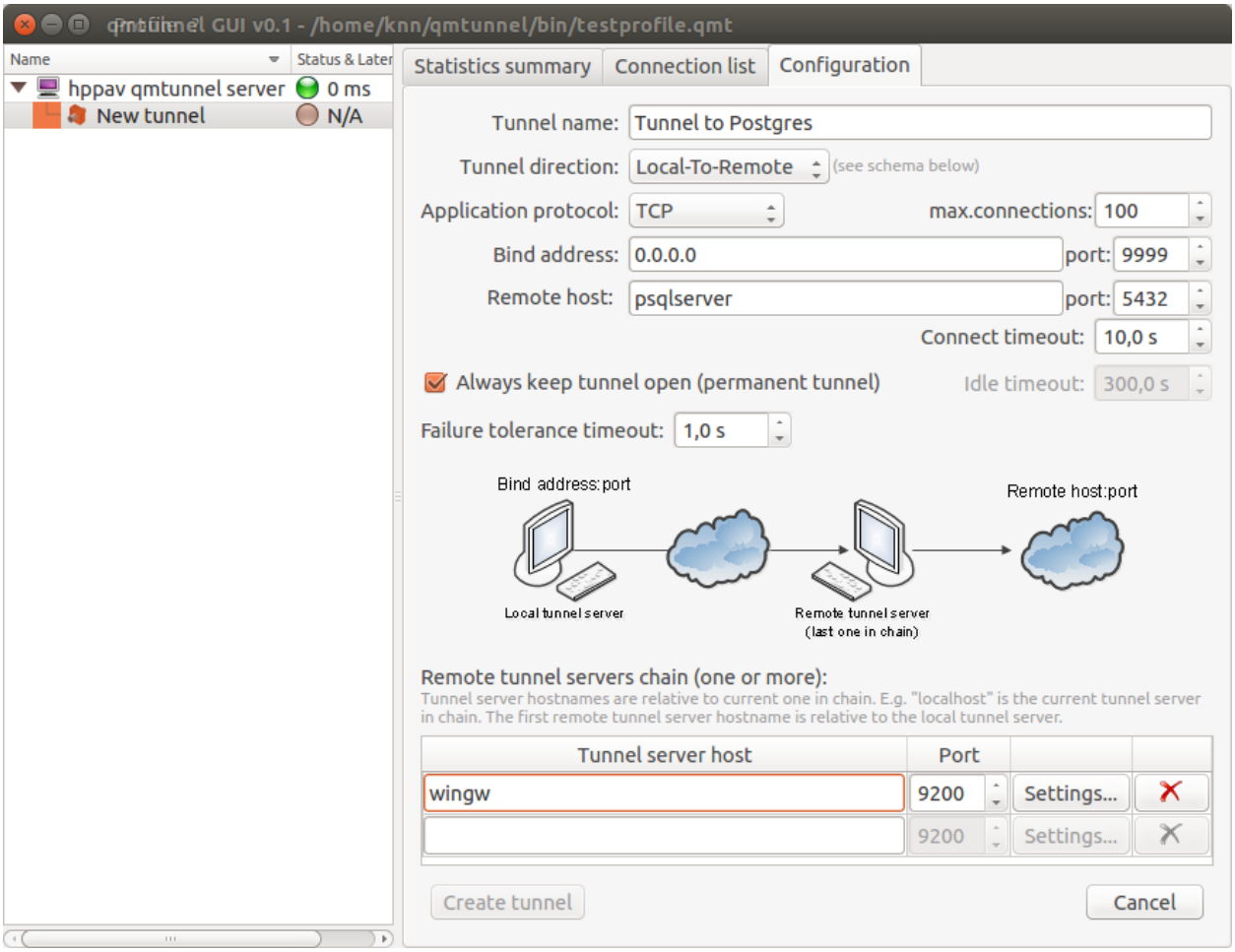
Press `Apply & Save`. Now the first qmTunnel server can connect to the second one.

Now let's create a simple local-forwarding tunnel.

Let's suppose that the second server is located in a remote network with a PostgreSQL server we need to access like this:



Return to `qmtunnel-gui`, right-click on our connected qmtunnel server and choose `Create new tunnel...`:

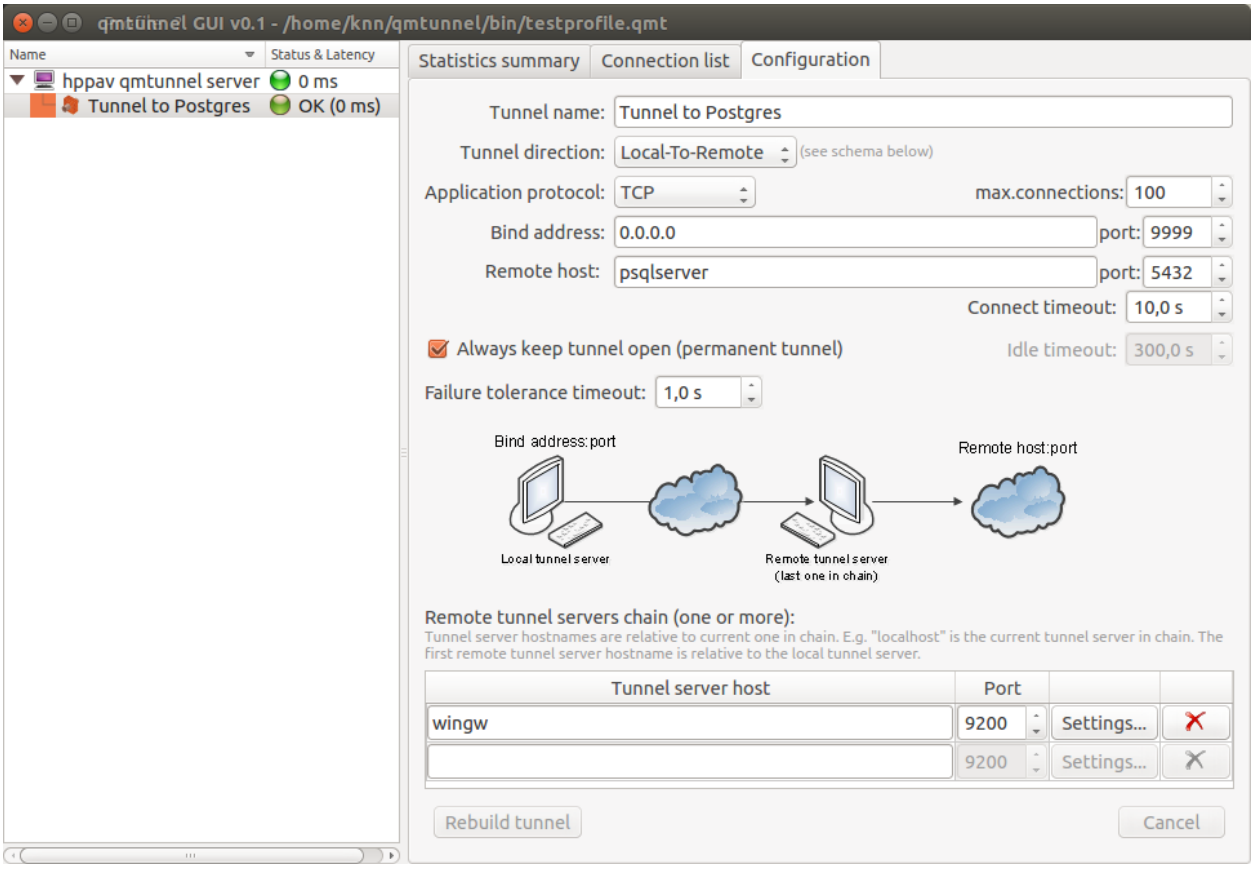


Fill up the following fields:

- **Tunnel name** - specify any tunnel name you want.
- **Bind address** - set to `0.0.0.0` if you want to share the tunnel with other hosts in your network, or `127.0.0.1` to make the tunnel available only from the current qmTunnel server host ("hppav").
- **port** (bind port) - set to any port available. `9999` was chosen for demonstration purposes. You can enter the same port as destination (5432) if it's available on qmTunnel host. This is the port your service/application client would connect to on the first qmTunnel server host.
- **Remote host** - set to IP address or domain name of destination service/application host. This is relative to the final qmTunnel server in chain and must be known on it. In the example above host "psqlserver" should be known on and available from "wingw" host. If you need to connect to the service/application located directly on the final qmTunnel host, you should enter `localhost` or `127.0.0.1` here.
- **port** (remote port) - set to final destination service/application port on remote host. This is where you actually want to connect to.
- **Always keep tunnel open** - Set this flag if you want this tunnel to be permanent and auto-reconnect when needed.
- **Remote tunnel servers chain** - add all qmTunnel servers in chain here except the first one. Be careful with domain/hostnames - they are also relative (should be known on and available from) each from previous one.

Then press the `Create tunnel` button.

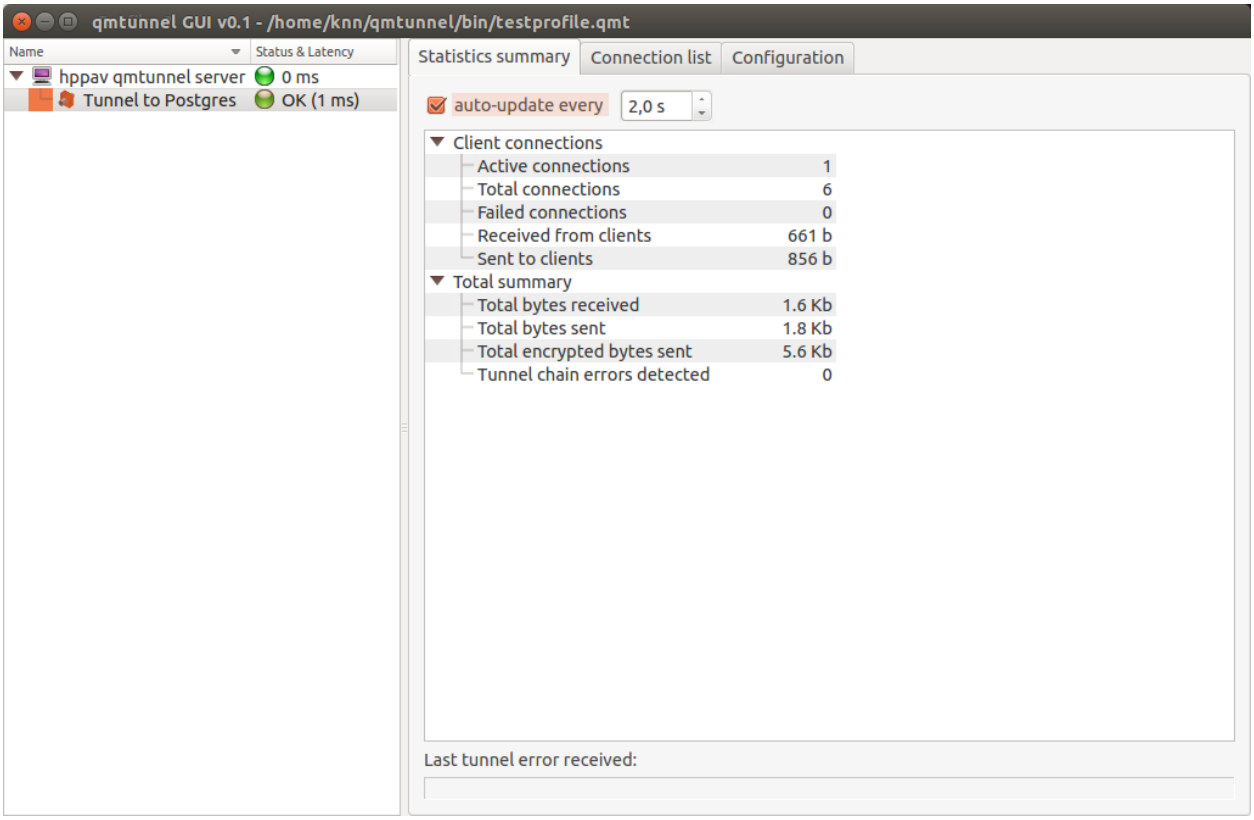
If everything is correct, you will see green indicator for your newly created tunnel:

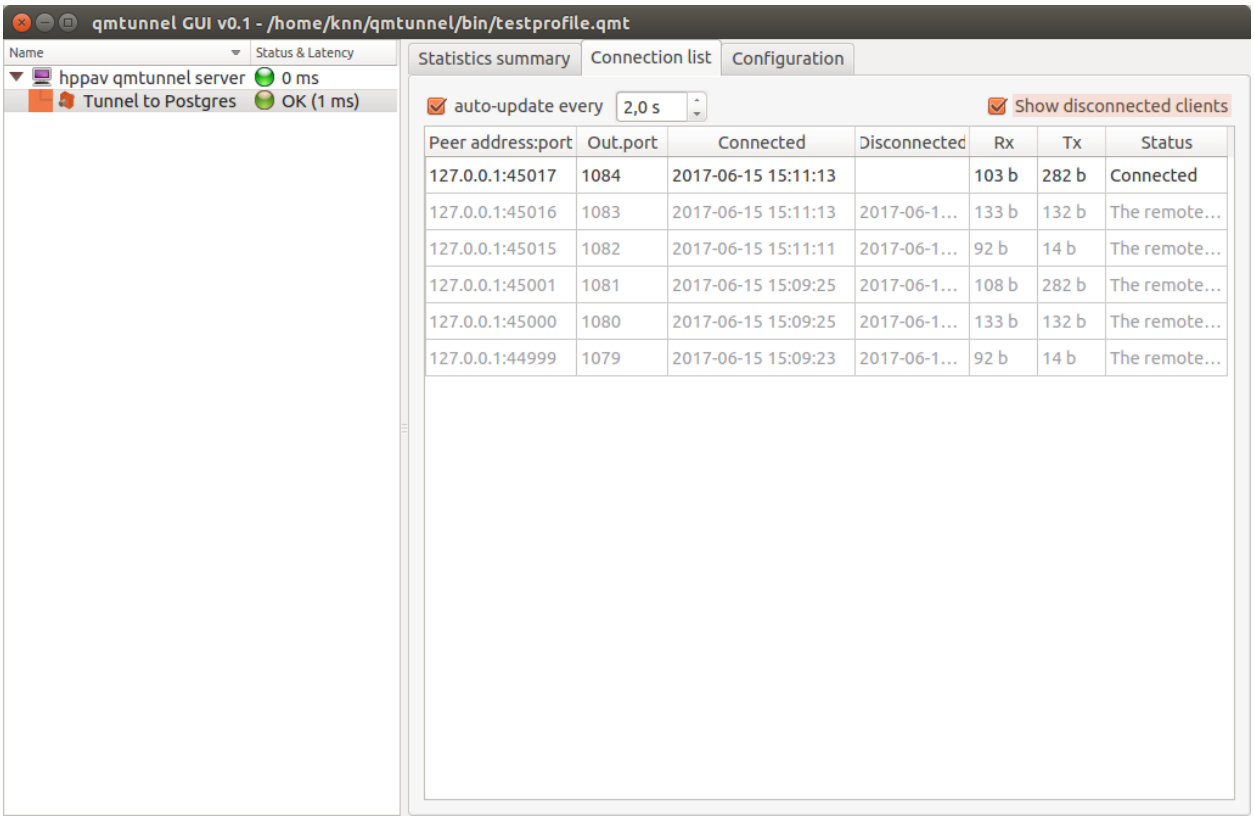


Now you can check the postgres connection to "hppav" which is transparently tunneled to "psqlserver":

```
psql -h hppav -p 9999 -U postgres postgres
```

You can now also monitor tunnel activity:


qmtunnel GUI v0.1 - /home/knn/qmtunnel/bin/testprofile.qmt
Name
Status & Latency
hppav qmtunnel server
0 ms
Tunnel to Postgres
OK (1 ms)
Statistics summary
Connection list
Configuration
auto-update every
2,0 s
Client connections
Active connections 1
Total connections 6
Failed connections 0
Received from clients 661 b
Sent to clients 856 b
Total summary
Total bytes received 1.6 Kb
Total bytes sent 1.8 Kb
Total encrypted bytes sent 5.6 Kb
Tunnel chain errors detected 0
Last tunnel error received:

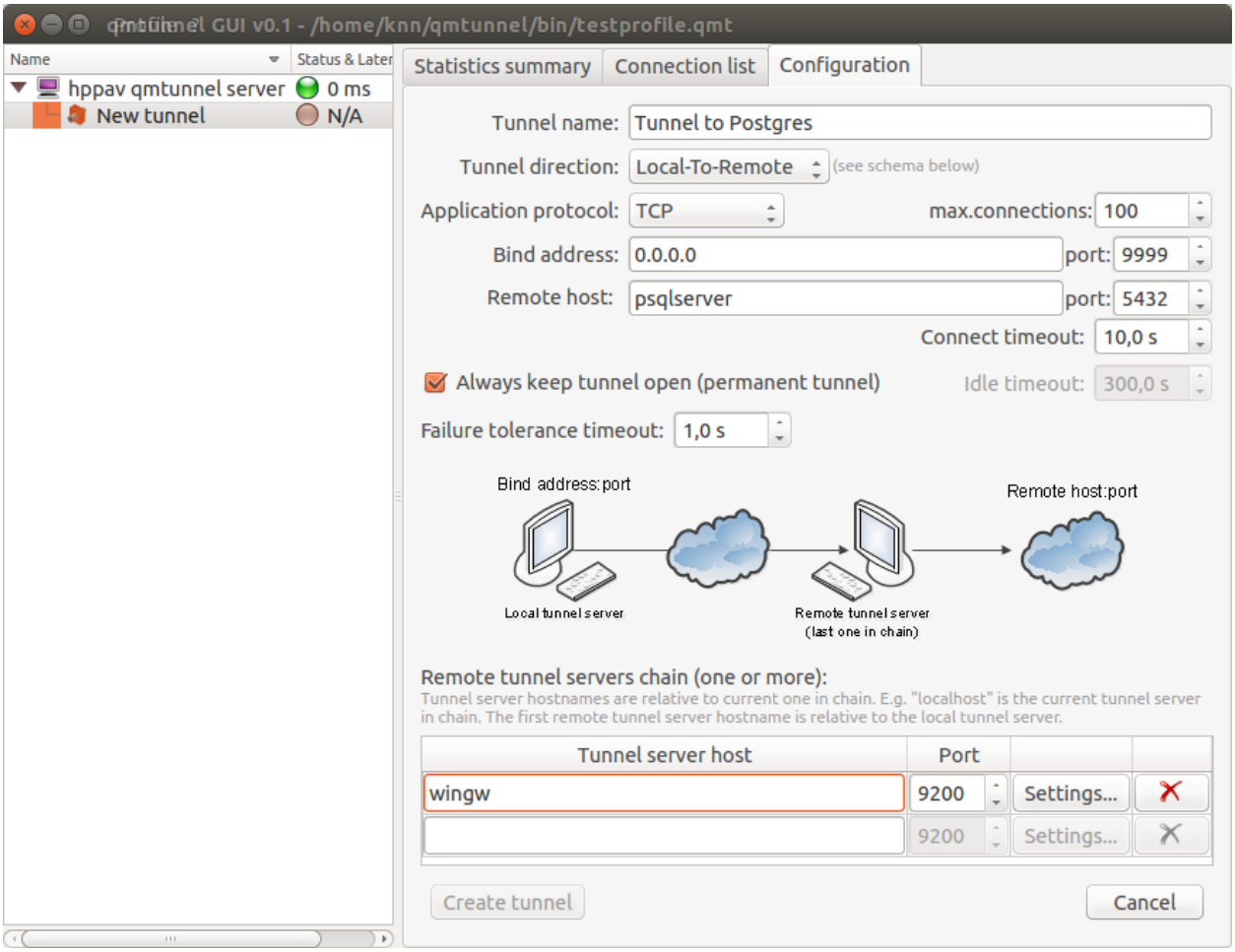

qmtunnel GUI v0.1 - /home/knn/qmtunnel/bin/testprofile.qmt
Name
Status & Latency
hppav qmtunnel server
0 ms
Tunnel to Postgres
OK (1 ms)
Statistics summary
Connection list
Configuration
auto-update every
2,0 s
Show disconnected clients
Peer address:port
Out.port
Connected
Disconnected
Rx
Tx
Status
127.0.0.1:45017 1084 2017-06-15 15:11:13 103 b 282 b Connected
127.0.0.1:45016 1083 2017-06-15 15:11:13 2017-06-1... 133 b 132 b The remote...
127.0.0.1:45015 1082 2017-06-15 15:11:11 2017-06-1... 92 b 14 b The remote...
127.0.0.1:45001 1081 2017-06-15 15:09:25 2017-06-1... 108 b 282 b The remote...
127.0.0.1:45000 1080 2017-06-15 15:09:25 2017-06-1... 133 b 132 b The remote...
127.0.0.1:44999 1079 2017-06-15 15:09:23 2017-06-1... 92 b 14 b The remote...

CHAPTER 3

Tunnel setup

Tunnel settings

When creating a tunnel, you can set up the following tunnel parameters:



- **Tunnel name** - specify any tunnel name you want.
- **Tunnel direction** - specify where to accept incoming connections from tunnel clients:
  - Local-To-Remote - accept incoming connections on the first tunnel server and forward traffic to the remote host of the last tunnel server in chain.
  - Remote-To-Local - accept incoming connections on the last tunnel server in chain and forward traffic to the remote host of the first tunnel server in chain.
- **Application protocol** - specify which protocol is used by the application client which traffic is to be forwarded.
- **max.connections** - maximum number of incoming application client connections which can be accepted.
- **Bind address** - set to `0.0.0.0` if you want to share the tunnel with other hosts in your network, or `127.0.0.1` to make the tunnel available only from the first (or the last in case of Remote-To-Local tunnel direction) qmTunnel server host.
- **port** (bind port) - set to any port available. You can enter the same port as remote destination if it's available on listening qmTunnel host. This is the port your service/application client would connect to on the first (or the last in case of Remote-To-Local tunnel direction) qmTunnel server host.
- **Remote host** - set to IP address or domain name of destination service/application host. This is relative to the last (or the first in case of Remote-To-Local tunnel direction) qmTunnel server in chain and must be known on it. If you need to connect to the service/application located directly on the qmTunnel host, you should enter `localhost` or `127.0.0.1` here.

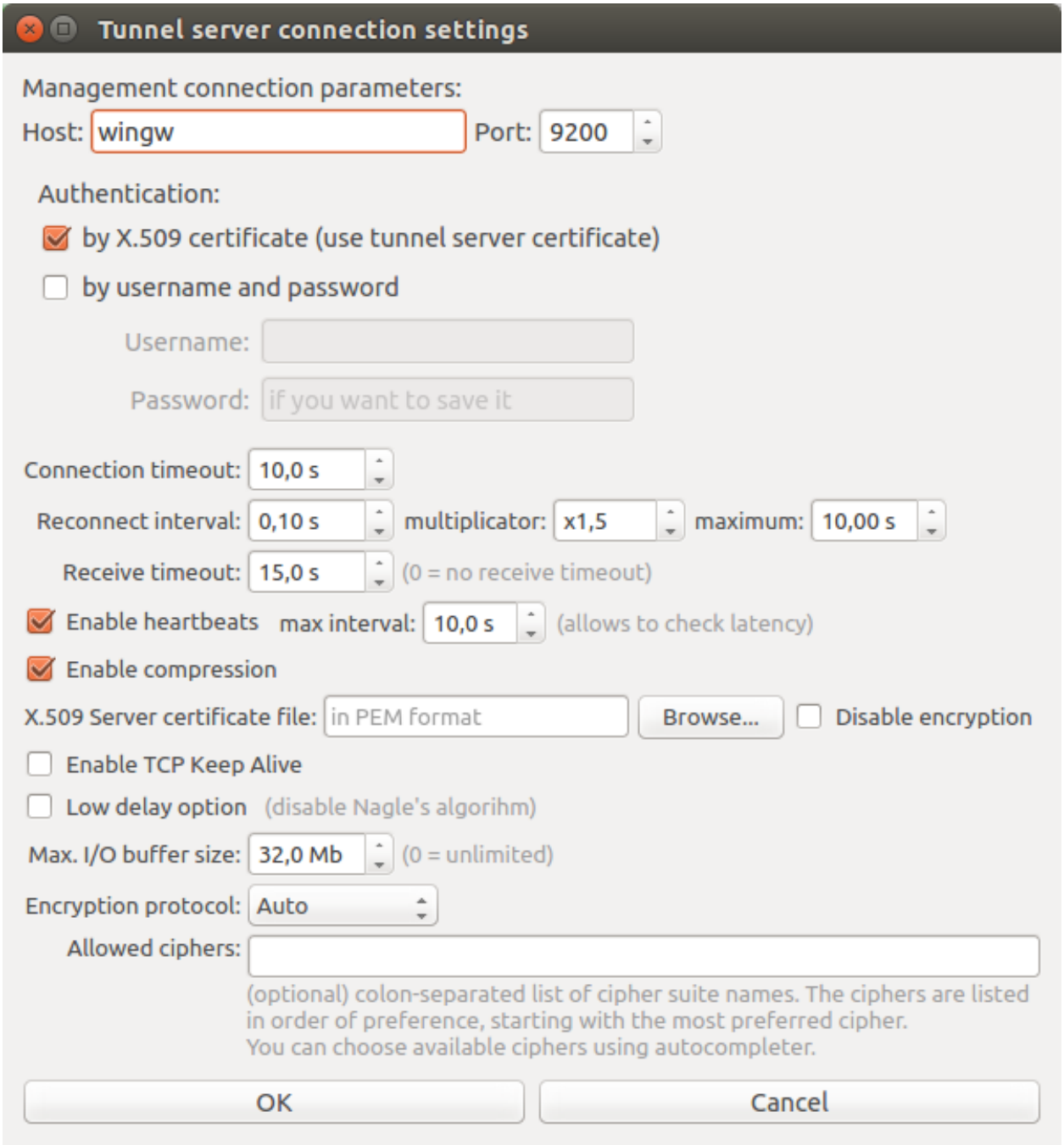
- **port** (remote port) - set to final destination service/application port on remote host. This is where you actually want to connect to.
- **Connection timeout** - application client connect timeout, used by qmTunnel server to establish outgoing connection to the application server.
- **Always keep tunnel open** - set this flag if you want this tunnel to be permanent and auto-reconnect when needed.
- **Idle timeout** - if there isn't any active application client connection for this time interval, consider the tunnel idle and disconnect. If new incoming application client connects after that, automatically re-establish the tunnel ("on demand" mode).
- **Failure tolerance timeout** - if any of the tunnel servers in chain disconnects, do not disconnect application clients during the timeout specified and try to silently re-establish the tunnel.
- **Remote tunnel servers chain** - add all qmTunnel servers in chain here except the first one. Be careful with domain/hostnames - they are also relative (should be known on and available from) each from previous one.

Tunnel servers connection settings

When creating a tunnel, you enter remote tunnel servers list (chain).

For every server in chain you can specify additional connection settings by pressing corresponding `Settings...` button in the same line.

The following window will then pop up:



- **Connection timeout** - Connection timeout when connecting to this tunnel server.
- **Reconnect interval** - When connection fails, try to reconnect after specified time interval.

  multiplicator - Multiply current reconnect interval after each failed connection attempt.

  maximum - Maximum value of current reconnect interval.
- **Receive timeout** - Consider connection failed if no data received in this interval. In pair with enabled heartbeats this allows to detect TCP silent connection drops.
- **Enable heartbeats** - Send small heartbeat packets if nothing else has been sent in **max interval** time interval.
- **Enable compression** - Enable traffic compression. Only application-level data is to be compressed (and only if

compressed data packet becomes smaller than uncompressed).

- **X.509 Server certificate file** - You can enter the path to qmTunnel server certificate file here and it will be checked after handshake. This allows to add additional security against MiTM attacks. Please note that at the moment this file path should be located on the first qmTunnel server host, so you have to edit it manually [will be fixed].
- **Encryption protocol** - Desired encryption protocol to use. Depending on Qt and OpenSSL version used, this may include: TLSv1.2, TLSv.1.1, TLSv.1.0, SSLv3, SSLv2. "Auto" means select most secure available protocol.

CHAPTER 4

Troubleshooting

If you're going to submit an issue or write to techinal support, please consider gathering the maximum of information. See below how to do that.

Debug logs

When started with `-debug` command line argument, both `qmtunnel-server` and `qmtunnel-gui` would write detailed log into logfile.

By default, logfile is `qmtunnel-server.log` and `qmtunnel-gui.log` correspondingly. You can change log filename with `-logfile FILENAME` command line option. For example:

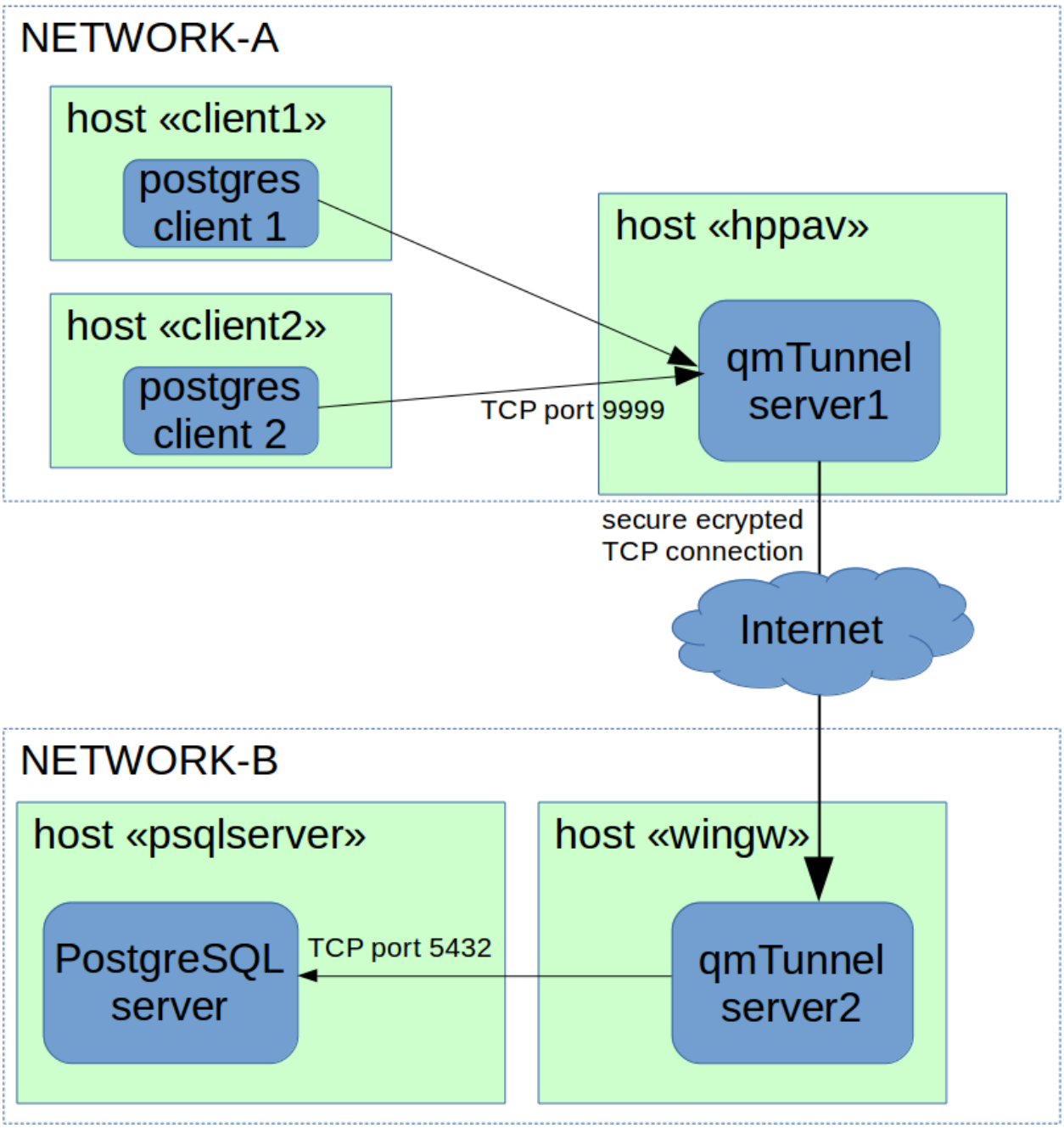
```
./qmtunnel-server -debug -logfile serverlog.txt
```

You can also limit log verbosity if you specify debug level (0 to 9):

```
./qmtunnel-server -debug 8
```

Draw your schema

If you realize something is going wrong way or you have hard times understanding the principles of forwarding and tunneling, start with drawing your network objects and connections you'd like to establish:



It's not perfect, but it might help you to better understand what you are doing.

Include you configuration

`qmtunnel-server` writes the configuration of the server to `qmtunnel-server.conf` file. You can change that with `-config` command line argument like this:

```
./qmtunnel-server -config serverconfig.json
```

The config file is written in JSON format and you can easily edit it manually if needed. You can also prepare such config file and copy it to another server if you wish.

Note: Config file doesn't contain any confedential information such as private keys.

Private keys should be stored in separate files in a safe place.

Ensure you include your config files along with debug logfiles and schema when submitting an issue.

CHAPTER 5

Building from source

To build qmtunnel, you need Qt >= 4.8 and OpenSSL <= 1.0.2 installed.

Note: If you wish, you can build them from source too:

- Qt: https://www.qt.io/download-open-source/
- OpenSSL: https://www.openssl.org/source/

RedHat/CentOS (7)

1. Install C++ development tools, Qt and OpenSSL:

```
sudo yum group install "Development Tools"
sudo yum install qt5-qtbase-devel openssl-devel
```

2. Download sources:

```
git clone https://github.com/karikhn/qmtunnel.git
```

3. Build:

```
cd qmtunnel/src/server
qmake-qt5
make
cd ../gui
qmake-qt5
make
```

4. Binaries will be created in qmtunnel/bin directory.

Ubuntu (14.04 LTS)

1. Install C++ development tools, Qt and OpenSSL:

```
sudo apt-get install build-essential git qtbase5-dev libssl-dev
```

2. Download sources:

```
git clone https://github.com/karikhn/qmtunnel.git
```

3. Build:

```
cd qmtunnel/src/server
qmake -qt=5
make
cd ../gui
qmake -qt=5
make
```

4. Binaries will be created in qmtunnel/bin directory.

Windows XP and later (32 bit)

1. Install Qt 5.6 with MinGW:

   http://download.qt.io/official_releases/qt/5.6/5.6.2/qt-opensource-windows-x86-mingw492-5.6.2.exe

2. Install OpenSSL 1.0.2:

   http://slproweb.com/download/Win32OpenSSL-1_0_2L.exe

3. Update PATH environment variable to include:
   - C:\Qt\Qt5.6.2\5.6\mingw49_32\bin
   - C:\Qt\Qt5.6.2\Tools\mingw492_32\bin
4. Get the latest qmtunnel sources from GitHub:

   https://github.com/karikhn/qmtunnel/archive/master.zip

5. Unpack, cd to qmtunnel directory and run:

```
cd src\gui
qmake
mingw32-make
cd ..\server
qmake
mingw32-make
```

6. Copy the following files to bin directory where qmtunnel-*.exe is located:
   - C:\OpenSSL-Win32\bin\libeay32.dll
   - C:\OpenSSL-Win32\bin\ssleay32.dll
   - C:\OpenSSL-Win32\bin\msvcr120.dll
   - C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\Qt5Core.dll
   - C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\Qt5Gui.dll

- C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\Qt5Network.dll
- C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\Qt5Widgets.dll
- C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\libgcc_s_dw2-1.dll
- C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\libstdc++-6.dll
- C:\Qt\Qt5.6.2\5.6\mingw49_32\bin\libwinpthread-1.dll
- Directory C:\Qt\Qt5.6.2\5.6\mingw49_32\plugins\platforms (actually only need platforms\qwindows.dll)

NEED HELP?

E-mail: support@qmtunnel.com

Note: Please be aware that individual-case direct technical support is delivered on a commercial basis.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search