User manual

（NAT Traversal Library）

Happytimesoft Technology Co., LTD

Declaration

Since the product version upgrade or other reasons, this manual will subsequently be updated. Unless otherwise agreed, this manual only as a guide, this manual all statements, information, recommendations do not constitute any express or implied warranties。

Table of Contents

Chapter 1 Files Description **4**

Chapter 2 Build Client **5**

2.1 Windows 5

2.2 Linux 5

2.3 Android 5

Chapter 3 Deploy **6**

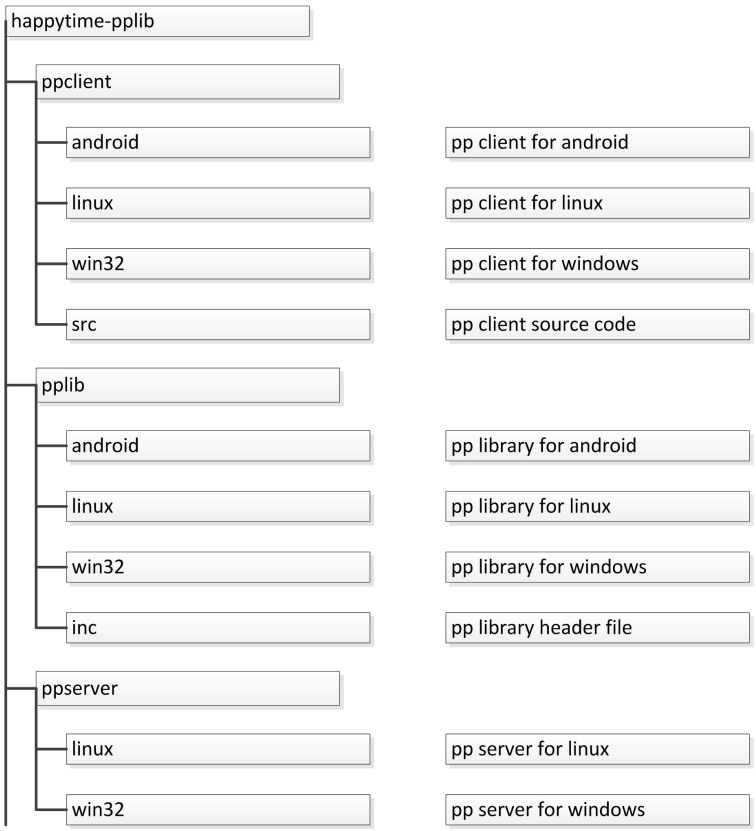
Chapter 4 Run & Test **7**

Chapter 5 API Interface **8**

Chapter 6 Example **10**

Chapter 1 Files Description

The download packet files structure as the following:



Chapter 2 Build Client

2.1 Windows

Dependent libraries: QT 4.8.0 and above

Use VS2008 open ppclient.sln to compile.

2.2 Linux

Dependent libraries: QT 4.8.0 and above

Use QtCreator open ppclient.pro to compile.

2.3 Android

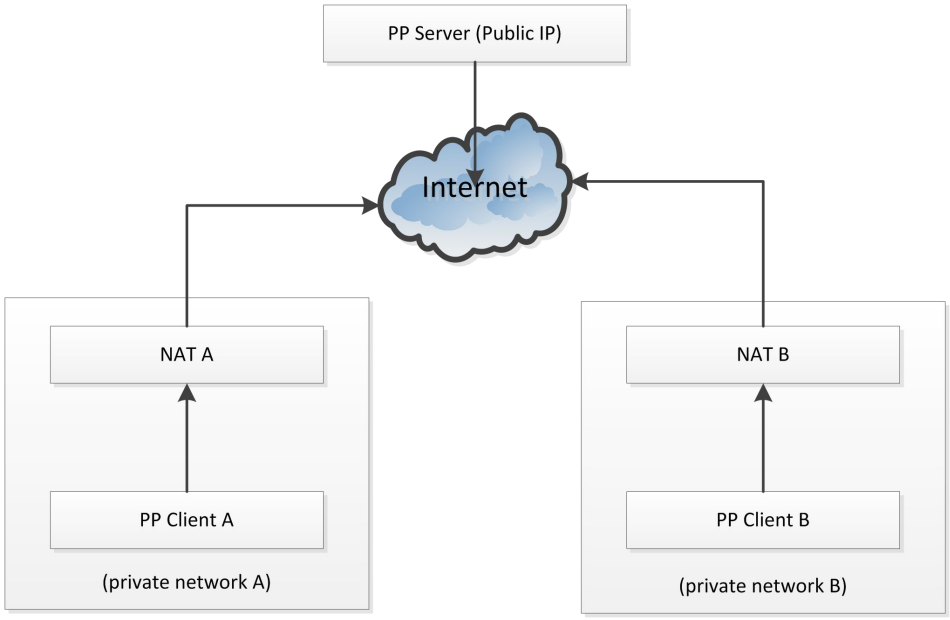
Dependent libraries: QT 5.2.1 for android and above

Use QtCreator open ppclient.pro to compile.

QT library download link: www.qt.io/download-open-source/

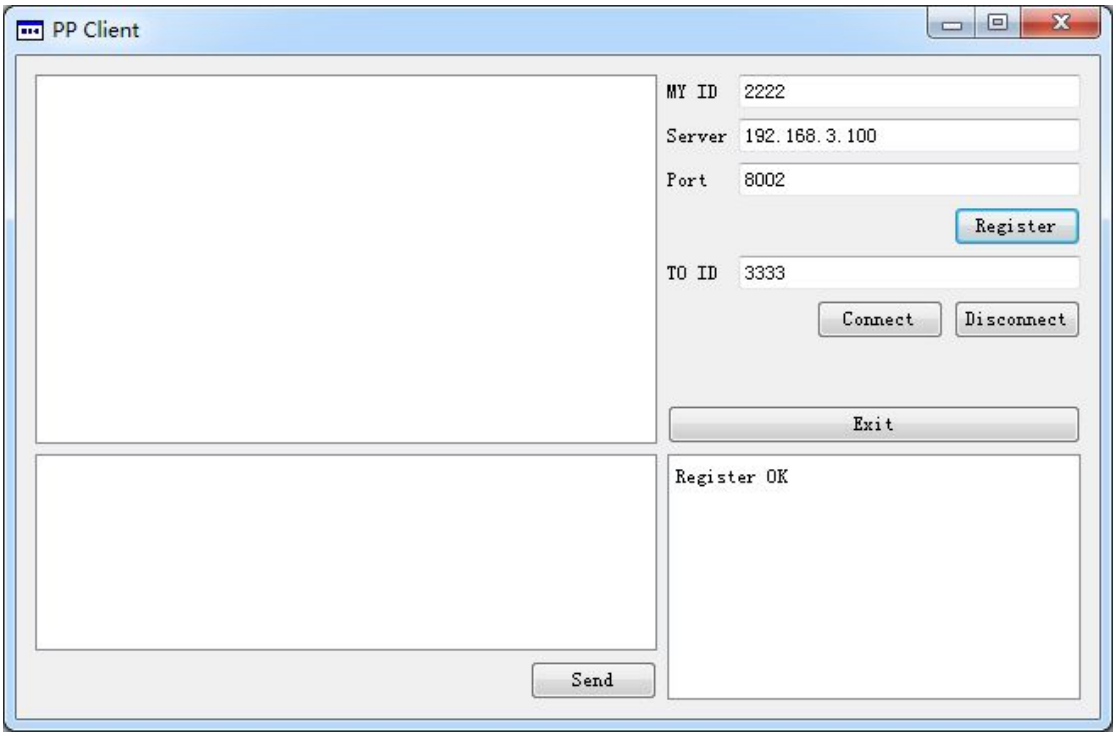
Chapter 3 Deploy

Network deployment topology as the following figure:



Chapter 4 Run & Test

On private network computer run the pp client, as the following:



Input the server ip address and port (the default port is 8002), set “MY ID” , click “Register” button.

Run another pp client on another private network computer, register to the server. Note the “MY ID” don’t the same.

Set “TO ID” to the peer, then click “Connect” button, the peer to peer connection will be automatically established.

Chapter 5 API Interface

PP library API interface description:

```
/* desc : set basic information
 * para :
 *    server : server ip or name
 *    port : server port
 *    user_id : the registered user id
*/
PPLIB_API void pp_info_set(const char * server, unsigned short port, const char * user_id);

/* desc : start pp stack
 * para :
 *
 * ret :
 *    successful return true, or return false
*/
PPLIB_API bool pp_start();

/* desc : stop pp stack
*/
PPLIB_API void pp_stop();

/* desc : start register
 * ret :
 *    successful return true, or return false
*/
PPLIB_API bool pp_start_reg();

/* desc : set event notify callback
 * para :
 *    p_func : callback function, the prototype please refer pp_exter.h PPNTFC
 *          p_userdata : callback data
*/
PPLIB_API void pp_set_ntf_cb(PPNTFC p_func, void * p_userdata);

/* desc : set data callback
```

```
 * para :
 *    p_func : callback function, the prototype please refer pp_extern.h PPDATAC
*/
PPLIB_API void pp_set_data_cb(PPDATAC p_func);

/* desc : try to connect user 'p_called'
 * ret : return the session handle
*/
PPLIB_API void * pp_make_call(const char * p_called);

/* desc : end connection session
 * para :
 *    p_sua : the connection session handle
*/
PPLIB_API void pp_end_call(void * p_sua);

/* desc : send data to each other use P2P channel
 * para :
 *    sua : the connection session handle
 *    p_data : data buffer
 *    len : data length
 * ret : successful return the sended data length, or return -1
*/
PPLIB_API int pp_send_data(void * sua, const char * p_data, int len);

/* desc : get the session remote user name
 * para :
 *    sua : the connection session handle
*/
PPLIB_API char * pp_get_remote_name(void * sua);

/* desc : get the session self name
 * para :
 *    sua : the connection session handle
*/
PPLIB_API char * pp_get_self_name(void * sua);
```

Chapter 6 Example

PP library API call example:

```
pp_start();

/* set callback function */
pp_set_ntf_cb();
pp_set_data_cb();

/* start register */
pp_info_set();
pp_start_reg();

/* establish P2P session */
pp_make_call();

/* send data */
pp_send_data();
...
pp_end_call();

/* establish other P2P session */
pp_make_call();

/* send data */
pp_send_data();
...
pp_end_call();

/* end of pp stack */
pp_stop();
```

The detail of the API call please refer to the source code of pp client