Subject: Re: Adding network proxy support to U++
Posted by Oblivion on Sun, 09 Nov 2014 15:56:42 GMT
View Forum Message <> Reply to Message

Hello Mindtraveller, Mirek.
Thank you for your suggestions. And I apologize I couldn't reply earlier, I was away for a while.

Here is what happened meanwhile:

After some toying with the network proxy idea for U++, I concluded that writing a monolithic, all-in-one network proxy class wouldn't be very effective (also imho, it would not be so upp-ish) way to implement it.

So I ditched the somewhat bloated, monolithic NetProxy class in favour of a modular approach, and abstracted a common interface and code for sync/async network proxies. This abstract class I called NetworkProxy. It contains all the necessary elements so far to succesfully implement most popular types of network proxy protocols. It has some advantages: Permits adding new/other sync/async network proxy protocols easily, relatively faster than NetProxy (due to less control code). Also we now have polymorphism: With this approach we can use a single Array object to contain a collection of different types of proxy instances if needed.

NetworkProxy abstract class currently has three derivatives with almost-identical interface: HttpProxy, Socks4Proxy, Socks5Proxy.

Here is the current "public" interface of NetworkProxy class and its derivatives (Note that SSL support and binding methods (in SOCKS4/5 proxies) are not yet implemented):

```
class NetworkProxy
{

public:
   enum ProxyType      { HTTP = 1, SOCKS4, SOCKS5 };

   NetworkProxy&       Attach(TcpSocket& sock)
   NetworkProxy&       Host(const String& host)
   NetworkProxy&       Port(int port)
   NetworkProxy&       Timeout(int ms)

   TcpSocket*          GetSocket() const
   int                 GetType() const

   bool                Connect(const String& host, int port);
   void                StartConnect(const String& host, int port);
   virtual bool        Do() = 0;

   bool                InProgress() const
   bool                IsSuccess() const
   bool                IsFailure() const
```

```
    String          GetErrorDesc() const

    NetworkProxy();
    virtual ~NetworkProxy();
};

class HttpProxy : public NetworkProxy
{

public:
    HttpProxy&      Auth(const String& user, const String& pass)
    bool            Do();

    HttpProxy();
    HttpProxy(TcpSocket& sock, const String& host, int port, const String& user = Null, const
String& pass = Null);
    ~HttpProxy();
};

class Socks4Proxy : public NetworkProxy
{

public:
    Socks4Proxy&    Auth(const String& user)
    bool            Do();

    Socks4Proxy();
    Socks4Proxy(TcpSocket& sock, const String& host, int port, const String& user = Null);
    ~Socks4Proxy();
};


class Socks5Proxy : public NetworkProxy
{

public:
    Socks5Proxy&    Auth(const String& user, const String& pass)
    bool            Do();

    Socks5Proxy();
    Socks5Proxy(TcpSocket& sock, const String& host, int port, const String& user = Null, const
String& pass = Null);
    ~Socks5Proxy();
};
```

As Mindtraveller suggested, I corrected the interface, but any further corrections, suggestions or

ideas are always welcome.

To give you further idea, here is the actual and complete working example code of async HTTP_CONNECT proxy (It simply connects to five different Ftp servers via a public http proxy and gets the server greeting):

```
#include <Core/Core.h>
#include <NetworkProxy/NetworkProxy.h>

using namespace Upp;

CONSOLE_APP_MAIN
{

 static const char *target_hosts[] = {
  "ftp.uni-erlangen.de", // Aminet ftp server.
  "ftp.mozilla.org", // Mozilla ftp server.
  "ftp.freebsd.org", // FreeBSD ftp server.
  "ftp.microsoft.com", // Microsoft ftp server.
  "ftp.kde.org",  // KDE ftp server.
 };

 Array<TcpSocket> sockets;
 Array<HttpProxy> proxies;
 SocketWaitEvent socketevent;

 for(int i = 0; i < 5; i++) {
  TcpSocket& socket = sockets.Add();
  socketevent.Add(socket);
  HttpProxy& proxy = proxies.Add();
  proxy.Attach(socket);
  proxy.Host("97.77.104.22");
  proxy.Port(80);
  proxy.StartConnect(target_hosts[i], 21);
 }

 while(proxies.GetCount()) {
  socketevent.Wait(10);
  for(int i = 0; i < proxies.GetCount(); i++) {
   HttpProxy& proxy = proxies[i];
   proxy.Do();
   if(!proxy.InProgress()) {
    if(proxy.IsFailure())
     Cout() << "Proxy connection failed. Reason: " << proxy.GetErrorDesc() << "\r\n";
    else
    if(proxy.IsSuccess()) {
     TcpSocket *socket = proxy.GetSocket();
```

```
     Cout() << "Succesfully connected via: " << socket->GetPeerAddr() << "\r\n";
     Cout() << socket->GetLine() << "\r\n";
    }
    proxies.Remove(i);
    break;
   }
  }
 }

 for(int i = 0; i < sockets.GetCount(); i++)
  sockets[i].Close();
}
```

There are some small quirks to iron out, but I am planning to upload the complete package next weekend.

Regards,
Oblivion.