Bypassing Restrictive Proxies

Version 1.00

Foreword

Anyone that knows how to get around a restrictive proxy system knows the painstaking process of explaining the methodology to friends. It's about time someone puts out a paper outlining basic restriction bypassing, and that is what we intend to do with this paper. However, a word of warning, if you downloaded this file thinking it's going to solve all your problems- quit now. This paper explains basic networking theory with regards to proxies, as well as bypassing methodology.

If you are simply looking for an easy way to get around the proxy without having to think or do anything, you'd be better off taking out your credit card and ordering an Internet account from another provider (maybe even one overseas) and log in paying international charges. Or worse yet, sign up for an AOL account.

Neither this file, nor the Flurnet website has any tolerance for script kiddies or people that wish only to bypass the proxy to get porn, as opposed to learning something new that could make your life more enjoyable, and making you a more educated person.

For those of you that are still unsure as to the purpose of this paper, its to enlighten the general public as to how a restrictive proxy system works. If you've never been unfortunate to get caught behind one, you'd probably not understand where this is coming from. In brief, some countries/organizations/administrators force clients to use a particular proxy (or proxy array) in order to censor particular information. This is moderately unethical on a corporate level, but far more so on a national level. Believe it or not, several countries (such as China, the United Arab Emirates, and others) have such systems implemented on a national scale. This paper will explain, briefly, how these systems work and what can be done to counter them.

Proxy System Explanation.

What a proxy is meant to do.

Originally proxies were developed to help speed up the Internet. Generally speaking, a proxy is a good thing, as it enables clients to retrieve websites stored on a network that's faster or more accessible. A proxy server is a sophisticated piece of software that manages a very large temporary cache of websites, constantly checking for updates. We will now discuss the utility of a proxy server, and how it theoretically is a good idea.

Typical non-proxied access:

A client types the URL he wishes to visit into his browser. The client's browser establishes a direct connection to the website they wish to view on port 80 and requests the pages, which are promptly delivered, parsed and displayed in the browser window. The website that's just been visited knows exactly who just visited it, because the connection was established directly.

Exactly 'how' direct is the connection, you ask? That depends on the geographical location of the website's server, the network that it is on, etc. If a client is located in Europe, naturally trying to connect to sites hosted in Europe will yield faster loading times then those hosted elsewhere- right? Often, data has to hop across thousands of miles, physically, before it reaches its destination.

Lets introduce the proxy to our mix now. A client requests a page by typing the URL into their browser, which is configured to use a proxy. The browser now requests the page from the proxy instead of the actual website. The proxy server will poll its cache for the files requested- if they exist, they will be delivered to the client. If they don't exist, then the server will download them itself before delivering them to the client, so as to speed up the process for the next person to request pages from the same site. The proxy server will constantly check sites for updates, refreshing them as often as possible.

What does this mean? It means that the client using a proxy server is never establishing a direct connection with the sites he or she wishes to access, a proxy server hosted on a super-fast backbone does it instead (who do u think can download faster, you or your ISP's proxy server? Duh!) Often proxy servers pass on variables such as the clients IP along to the website for logistical purposes, amongst other things.

On the topic of logistics, its important to mention that proxies often log all requests, and naturally a direct connection is being established to a proxy- does that affirm your fears? Your ISP has the ability to find out **EXACTLY** what sites you are visiting, and when if they care to dig through logs. Scared yet?

How a proxy can block websites.

Before getting into this topic, we have to get through a little networking theory. When a browser requests a page directly, it first resolves the URL's hostname (www.flurnet.org) to its IP address. Next it opens a connection to the corresponding IP at port 80- which has to be opening in order for a website to exist. We will be talking about IP addresses and ports frequently- its important to know that typical web servers operate on port 80, and proxy servers on port 8080.

An ISP, just like a private or corporate network, has the ability to set certain rules about connections incoming or outgoing. A common one for isp's with restrictive proxies is to block all outgoing activity on port 8080, unless its going to a specific destination (ie. restrictive proxy server). Why do they do this? To keep clients from being able to establish any connections to machines on port 8080 anywhere- hence disallowing clients to connect to any proxy servers but their own. To make matters worse, they will also block absolutely all traffic on port 80- to prevent clients from being able to establish direct connections to websites. The next logical step would be to configure the proxy server to return an error when a client attempts to retrieve a site matching some criteria (like keywords, specific urls, etc.)

Just to seal the deal and make it as hard as possible for clients to get through the system, ISPs then block all ports they think may contain proxies running on non-standard ports. One such port is 3128, another is 10080. Some ISP's in the Gulf deny access on many more ports that offer services such as VoIP (preventing clients from being able to place calls over the Internet, so as to avoid creating competition for telecommunications customers- YEAH, I'M TALKING ABOUT YOU ETISALAT!) Bastards.

Now, clients are forced to utilize a proxy server that restricts their browsing to sites deemed acceptable by proxy server administrators. Clients cannot connect to websites directly, and cannot use any other services that utilize ports 80, 8080 or other blocked ports.

But at least we have a nice fast connection to our proxy server, which has a nice fast internet connection- right? Not quite. As more and more clients connect to the network, a proxy server will experience more strain from many simultaneous connections, often more clients are requesting pages then the proxy server can handle. This is called a bottleneck. Once an ISP mixes broadband (such as DSL or cable internet) to the equation, a huge mess is created. Proxy servers must be connected to faster lines, must have more hard drive space for larger caches, more RAM to allow more simultaneous clients, better infrastructure so as not to cause a network bottleneck, etc. All this requires a steady flow of financing, and who really wants to lose potential profit these days? On top of having restricted Internet, we now are dealing with SLOW restricted Internet. If you are depressed because you realize that this happening to you right this second, don't worry- we will be discussing ways on bypassing these restricting systems very soon.

Transparent Proxies

To make client configuration simpler, service providers might install a transparent proxy. This is often the case with restrictive broadband access. A transparent proxy is configured on the gateway level. When a client with no proxy configured tries to connect to a website, the ISP wont drop packets, instead they will be passed through a proxy at the next hop. This causes a lot of confusion amongst more advanced users- as they may be led to believe that they are browsing without a proxy- when in fact a restrictive proxy is in full effect.

An easy way to test a transparent restrictive proxy is to try establishing a direct connection to a censored website using a telnet client on port 80. Once a connection is established, issue a 'GET' command (type GET and press enter). Parsing the html output should display the standard proxy error. This means that even a direct, proxy-free connection is still going through a proxy…

On a similar note, it's important to mention that often ISP's don't just use a single proxy server. Transparent proxies will be different from dialup proxies and so on. There are usually arrays of proxy servers- Etisalat has at least 10 proxy servers for its various customers for example. Oddly enough, some proxies are more restrictive then others- and often particular proxies bottleneck or lag. Since all the proxies are within the clients IP segment, any can be used. If an intelligent client notices that the transparent proxy is being slow, changing to a dialup proxy can help dramatically. Many tools are available for determining proxy latency.

While on the subject of proxy arrays, another point worthy of mention is censorship accuracy. Often the various proxies within an array have different rule sets or are misconfigured- some proxies may allow old URL encoding (hex/bin discussed later) whereas others may not have censored pages that are censored on transparent proxies. Experimentation is useful, and documentation is more so.

Restrictive ISPs are yet to devise a system that synchronizes the access granted by various proxies, clients are recommended to take advantage of this 'vulnerability' while it lasts.

Staying Ahead & Intellectual Evolution

Hostname resolution

Now that the theory part is through, let's get into some history. In this section we'll discuss primitive methods to bypassing restrictions, most of which will probably not work on your network- but they might- who knows? (The sure-fire ways come later.)

What determines whether or not a site should be proxied? Once upon a time, all that stood between a client connecting to a site or not was a database. There was a master list containing the address of every single website deemed unsuitable for clients to view, and that list didn't contain any IP addresses. As we mentioned earlier, every hostname resolves to an IP address- what our ISP had failed to realize was that every IP reverse-resolves back into a hostname. Thus, using simple tools downloaded from the Internet (such as SamSpade, http://www.samspade.com) we were able to resolve the IP addresses of sites we wished to visit, and feed them to our browser instead of the hostname. Here is an example:

> Resolved www.flurnet.org (hostname) to 192.168.1.1 (IP)
> Visit http://192.168.1.1 and enjoy Flurnet, unproxied.

ISP's caught on to this pretty quickly, as all they have to do is look through their proxy logs and see that lots of clients are connecting to sites using IP's instead of hostnames. A patch was written- forcing the proxy server to always reverse-resolve IP addresses and compare them to the database: effectively shutting down our system. By now you probably understand what 'Staying Ahead' is all about. The object is to bypass a proxy system without tipping it off as to how we are getting around it- but back then we had no idea.

IP Conversion

After the reverse-resolution patch was implemented, we quickly discovered other interesting things one could do to 'trick' proxy servers. By tricks, we mean arithmetical conversions of course. An IP address is four numbers, separated by dots. No numbers ever go higher then 255. Each of these four numbers is in 'decimal' format (base 9). Computers only really care about 1s and 0s, so we figured if we could tell it to decipher the IP address as, say, an octal or hexadecimal value, we'd be able to trick the proxy once more. Naturally, it worked.

Some examples:

Hexadecimal

```
http://www.fakesite.com
->    194.170.168.35

035 Dec = 23 Hex
168 Dec = A8 Hex
170 Dec = AA Hex
194 Dec = C2 Hex

http://0xC2.0xAA.0xA8.0x23
```

Octal

```
http://www.fakesite.com
->    194.170.168.35

035 Dec = 043 Oct
168 Dec = 250 Oct
170 Dec = 252 Oct
194 Dec = 302 Oct

http://0302.0252.0250.043
```

Dword

```
http://194.170.1.6
      -> http://3265921286
      -> http://218014286086
```

Convert each segment of the IP into hexadecimal format:

```
      194 = C2, 170=AA, 1=1, 6=6
```

Next put them all together, putting a zero behind any 1-digit number- you should always have an 8 digit hex number at this point:

```
      =     C2AA0106
```

Convert this 8 digit hexadecimal integer back into decimal, and that's your DWORD:
`3265921286.`

To make things even more complicated, you can add any multiple of 256^4 to this number and it will still work, this is because, quite simply, the last 8 hex digits will always remain the same- everything else simply gets ignored.

i.e.

```
      3265921286 + (256^4) = 7560888582
      7560888582 = 1C2AA0106 Hex.
      3265921286 + (2*256^4) = 11855855878
      11855855878 = 2C2AA0106 Hex.
      3265921286 + (25*256^4) = 218014286086
      218014286086 = 32C2AA0106 Hex.
```

Other ways included binary conversions and IP permutations. Flurnet released a handy utility to its members at the time that would resolve any hostname, and calculate the IP then send it back to the browser. Other similar utilities began popping up- and before long a browser modification was available, converting and calculating addresses on the fly. It didn't take long before administrators noticed these strange addresses in their proxy logs. Soon, a new patch was installed, and this proxy-confusing bypass method became obsolete (or so was the case with my old ISP.) This method may possibly still work on your restrictive network, as patching something like this would be extremely complicated. Plus it's a pretty neat thing to do, your colleagues will certainly be impressed. Performing the calculations can be accomplished on any scientific calculator, or better yet, the windows calculator in scientific mode. Just enter the numbers in decimal format, then click on the radio button labeled 'hex' or 'oct' and voila, instant conversion. For the extremely lazy, a few clicks and you could download a tiny application flur wrote to do it for you. Check the file archives on www.flurnet.org.

Web based CGI Proxies

So, math isn't your thing? Chances are you've seen one of these in action before. CGI proxies began popping up in a strange fashion- originally they were tools that would play with pages- for example- you type the address of your favorite website, and the CGI script would retrieve it, flip everything, then output it backwards for your amusement. People quickly realized that these tools were a great way to get past their restrictions, and so these scripts quickly evolved into fully blown web-based proxy servers.

What is a Web-based proxy anyway? Web based proxies, commonly referred to as CGI proxies are websites with an intelligent backend. Basically, it's just a website where a visitor can type an address into a text box on the site for the server the site is hosted on to download, then display. Let's evaluate this in a little more depth:

A client tries to connect to web-proxy such as anonymizer (www.anonymizer.com). The request goes through the restrictive proxy server, and if allowed, the client is able to view the web-proxy site. In layman's terms, you can use these to view proxied sites, but only until the restrictive proxy server administrators discover the site (which often happens quick, because everyone starts using them- making them easy to spot in logs.) Once the site is added to the restricted sites list, clients will need to find a new cgi proxy.

To make matters a little more complicated, CGI proxies usually create temporary mirror copies of the requested sites. For example, if you were to request that www.cgiproxy.com retrieve www.flurnet.org for you, the link you would receive will look something like this:

http://www.cgiproxy.com/mirror.cgi?site=www.flurnet.org

What is important to notice is the fact that the destination address is in the URL. Being bastards, the restrictive proxy server administrators decided to enforce bans on keywords or URL's embedded within other URLs. This means that even if cgiproxy.com wasn't restricted, the proxy server would deny access to the site because it contained a particular address within the URL. The same can be done with keywords on a site, or within a URL- often the case with restricting particular searches (it's not actually part of the search engine, its just checking out the URLs that your browser is requesting.)

Another point worthy of mention is that CGI-Proxy administrators can see what sites you are visiting. Not that it matters much, but it could be in violation of your privacy.

Modern CGI-Proxy systems have gotten sophisticated enough so as to encrypt a URL to avoid detection by a restrictive proxy. This does ensure that the system will operate provided that its not restricted. An example of an encrypted CGI-Proxy's url:

http://www.cgiproxy.com/mirror.cgi?site=AS9859FJFA8t592352AFJ59205jSA

CGI Proxies are the most popular way to get past the proxy, and many are operational today. The difficult part is finding them before they are restricted. Apart from searching on google, a good suggestion would be to consider running your own website with a private CGI proxy that operates just for you. Wouldn't that be great? Moving along, but staying on topic, lets have a look at some options as to ways to get past proxy restrictions using a web hosting or UNIX Shell account.

Current theory
SSH (-R) & Redirection

Finally, the big guns. This section deals with modern methods to get around proxy restrictions. In the section above we mentioned web hosting and shell accounts- let's elaborate a little on that before we move on.

Most web hosting companies have servers that run some flavor of UNIX or Linux. Often, when a new account is created, the web hosting company gives you access to the machine via telnet (or ssh- an encrypted terminal client) as well as ftp and web. Many people don't ever bother connecting to the shell using a terminal client, because, well, what is there to do on a Unix machine hosted somewhere far away?

SSH uses port 22, and telnet uses port 23. These ports are open on most restrictive networks, because they are very popular amongst system administrators and advanced users. Once connected this remote machine, we can instruct it to do many things for us- namely we can use a 'redirector.'

Remember how we discussed earlier that port 8080 and 3128 (amongst many other ports) were blocked? Well, what if we were to find a proxy on an unblocked port? You could open up your browser, go to the proxy configuration screen and enter this uncommon proxy and port- and you'd have unrestricted Internet access which DOESN'T go through the restrictive proxy.

We will use our shell account to seemingly create a proxy on a port of our choice, however, instead of wasting plenty of time configuring a proxy server, we're simply going to redirect some traffic. Considering that the remote machine has no restrictions, it should be able to connect to any proxy on port 8080 or 3128- the only problem is that the machine is all the way in the United States, and you are all the way, well, here. No problem! We can instruct the remote machine to 'redirect' all traffic from a certain address/port to its own address at a port which we can specify.

Client → Shell (2600) → Proxy (8080)
Client ← Shell (2600) ← Proxy (8080)

Redirectors often come built into Unix/Linux operating systems. This is a breakdown of the command line syntax:

ssh -R 2600:proxy.com:8080 shell.net

This command will redirect 'proxy.com:8080' to itself (shell.net) on port 2600. After you enter this command into your shell account (with a working proxy of course) you should be able to put the shell's address into your browsers proxy settings along with the port you are redirecting to (2600, in our case) and you'll be able to browse the net without restriction. This is because your requests now go directly to your shell account on an uncommon, unblocked port. The shell passes the request along to the proxy server you

specified, which sends data back to your shell- intelligently enough- the shell forwards the data back to you.

If you understood that, then between your eyeballs is a method for bypassing proxies so powerful, that no ISP could ever stop you. If you wish, you can redirect traffic to ports used for other services (provided your shell provider permits it)- no self-respecting censoring ISP would consider blocking, say, SMTP (port 25) or FTP (port 21). We can always create tunnels to proxies on those ports.

Shell accounts have certainly been purchased for less important tasks, so if you are serious about your browsing maybe its time you got yourself one. They aren't too expensive- and it will teach you the basics of a linux/unix environment. Shell accounts can be as cheap as $3 per month, and there are a lot more then just bypassing proxies that you can do with them.

Let's run through it step by step. To join us, you will need a shell account (search the web) and a list of proxies running on common, blocked ports (search the web.) Assuming you have both, this is what you'd do.

1. *Connect to the shell account using telnet or an ssh client (such as puTTY.)*
2. *Issue a redirection command:*
   ssh -R 2600:proxy.com:8080 shell.net
3. *Change your browsers proxy settings from the defaults to the shell's address (in this case, shell.net)*
4. *Enjoy browsing uncensored Internet.*

You must note however, that if you close the connection to the shell, you will lose your redirection tunnel- so that must stay open. Also, if you trail an '&' after your command, you will be able to use the shell, as the tunnel will be opened in the background (it will close when you log out or close the connection to the shell.)

SSL & Port 443

SSL, or 'Secure Socket Layer' is what is responsible for creating that yellow 'key' icon in most browsers, it is what makes a 'secure' website secure. It does this by establishing encrypted communications over a direct connection on port 443. No ISP would be ignorant enough to block direct connections to this port (as it would be defeating the purpose of the SSL features) – how is this relevant you ask? Your ISP probably tries tricking you into configuring proxies for services that you don't need. Check your browser settings, are you using automatic or manual proxy configuration (if you are using automatic, unset it immediately.) If you are using manual proxy settings, you should rest assured that FTP port 21 is open, so no proxy is required for FTP. Security (port 443) is open so there is certainly no need to use a proxy there. Actually, the only proxy you should have in your manual proxy settings is HTTP- and hopefully its not the one your ISP instructed u to use by now ;) !

The point this subsection is trying to make is quite simple, if no proxy is configured for the secure socket layer, a direct connection can be established. The only problem is that very few websites have SSL, and those that do rarely have their entire websites mirrored. But it's useful for techies like us monkeys at Flurnet- because we know that if we ever get on restricted lists, all we'd have to do is enable SSL and link up our whole site. ISP's wouldn't be able to restrict that too easily, and they'd have a far more limited log of your activities on our site.

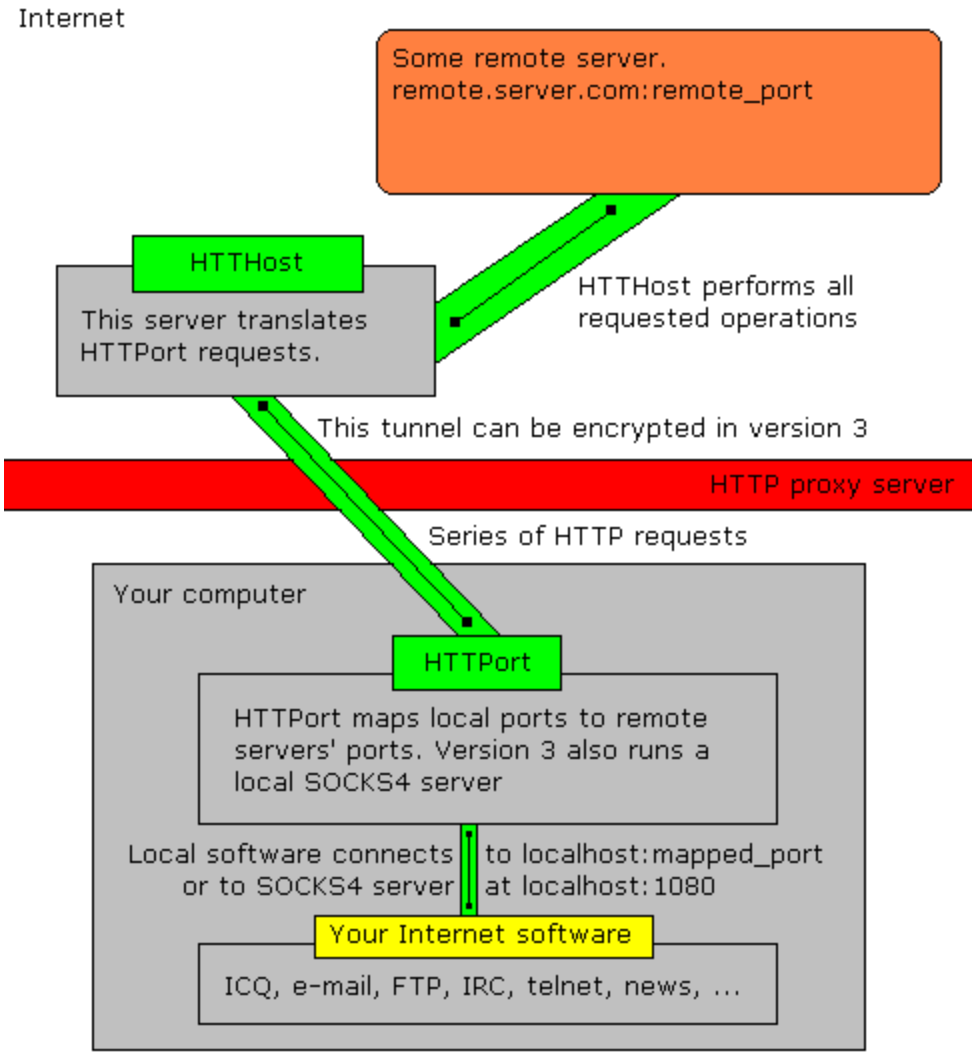
If you are a website administrator, you may want to consider experimenting with SSL as it's almost a guaranteed way clients will be able to connect to your site directly.

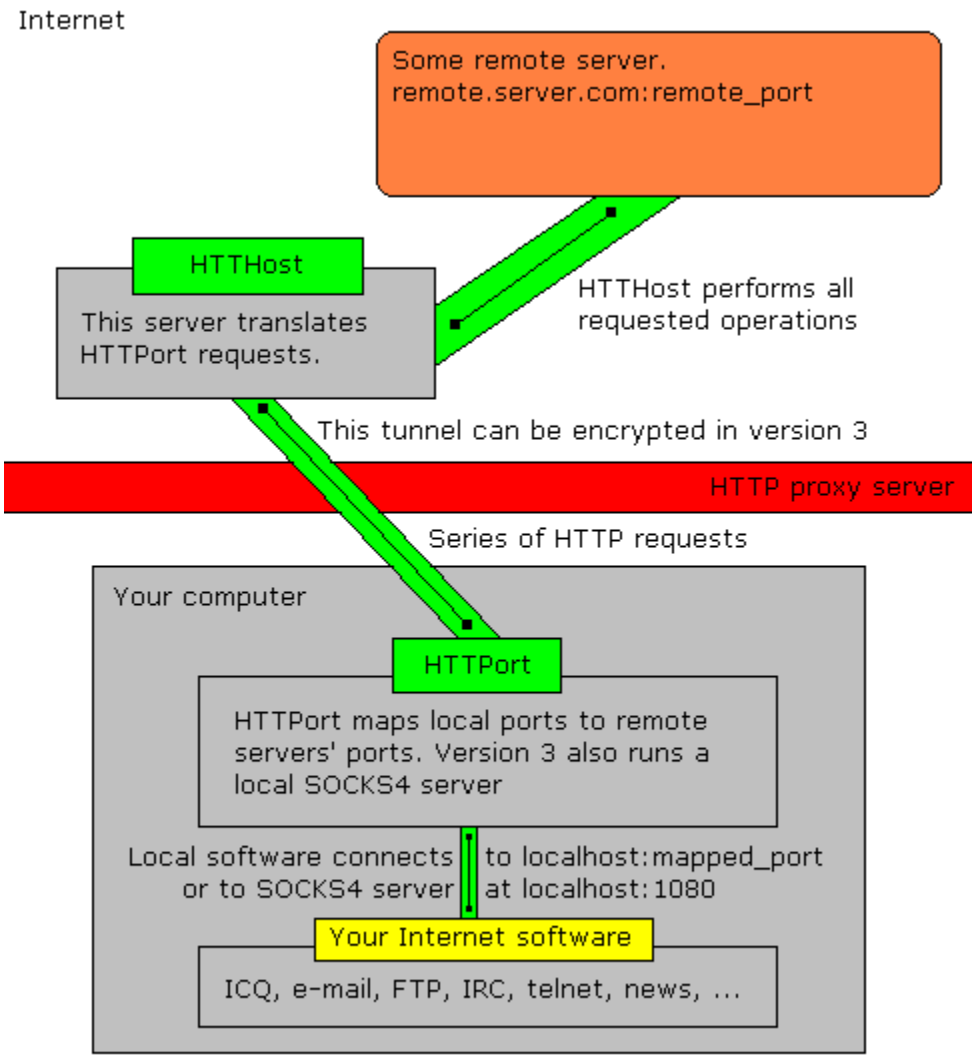
HTTPort & Connect Proxies

HTTPort allows you to bypass an HTTP proxy, which is blocking you from the Internet. With HTTPort you may use the following software (just a sample list, not limited to !) from behind an HTTP proxy: e-mail, IRC, ICQ, news, FTP, AIM, any SOCKS capable software, etc. etc.

The basic idea is that you set up your Internet software in such a manner, that it considers your local PC to be a remote server it needs. This is where HTTPort enters. It intercepts connection from this software and runs the connection through the proxy - this is called a tunneling.

HTTPort performs tunneling using one of two modes. The following schemes will give you the idea:





The reason that this works is because some proxies are configured (misconfigured?) to allow clients to establish raw connections through a proxy- a good idea would be to use a restrictive but connectable proxy to create a tunnel to a non-restrictive proxy. Connectable proxies can also be used to connect to IRC (spoofing your IP as that of the proxy) however, most IRC servers have extensive lists of proxies, and scripts to check and make sure that a proxy server isn't running before allowing a client to connect. Less popular IRC servers will still allow you though.

Some proxies don't allow inbound connections except from IP addresses within their own subnet- this is great for IRC, because the IRC server wont be able to determine that the IP is running a proxy (as it wouldn't be allowed to use the proxy anyway.)

Using tools like HTTPort, clients can over-ride any proxy setting. The only drawback is that one of the restrictive proxies must have a 'connect' feature. An invaluable resource is www.antiproxy.org, as it provides lists of public proxies, and the ability for members (free membership) to check proxies for 'connect' features. It's also a good site for further reading if you are still interested.

Proxies on Uncommon Ports.

The fastest, easiest method- saved for the end of course. ISPs rarely set up their network to only allow connections on certain ports (as in, 'deny all except'), instead the rules usually allow connections to all ports with exceptions ('allow all except').

There are several lists of ports blocked by ISPs floating around the web. The ISP worthy of mention in this section is Emirates Internet, as they allow connections to port 8000 (a relatively common proxy port) for some undisclosed reason. Let it not ever be said that stupidity isn't an asset.

What does all this mean? You sure are slow. Heh. Get a list of working proxies on port 8000 (they go down pretty frequently) and simply replace your current HTTP proxy settings with any address running a proxy on port 8000 (don't forget to change the port from 8080, your local providers port, to 8000- that which the new proxy operates on.)

This should give you the fastest connections, specially if you can find a fast proxy. Search the web, and be sure to search the extensive databases on AntiProxy (the site supports SSL too, so if its blocked by your provider be sure to check https://www.antiproxy.com (yes http**S**- this instructs your browser to connect to port 443 instead of 80).

As for people in KSA amongst other countries, your ISP doesn't allow connections to port 8000, do don't even bother trying.

Conclusion.

You are now qualified to choose a proxy bypassing method that you find suitable. At the time of publication, Flurnet operates a small CGI proxy (which wont display images, and doesn't encrypt URLs) for clients that want to dig up more information to use. You can find it in our 'Online Tools' section, under the 'Services' submenu.

If you have any questions, or noticed any errors or inaccuracies in this file- or if you have some information to add (other methods/research/etc) then please email me at flur@flurnet.org. However, **DO NOT EMAIL ME** if you skimmed these pages briefly, thought it looked too complicated and would prefer if I sent you step by step instructions. Any e-mails directed to me involving questions addressed in this paper will simply be ignored.

Your options now are:

(a) You understood this material and are ready to give it a shot
(b) You didn't understand this too well and are going to re-read it or do some more research.
(c) You are an idiot that doesn't to even be online.
