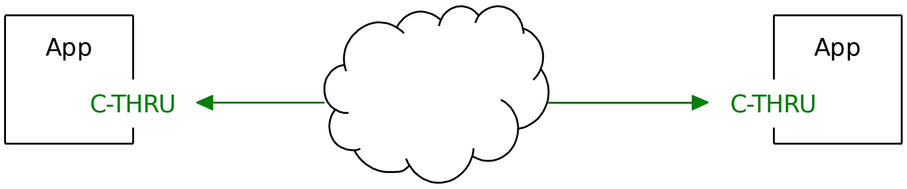
C-THRU™ Functional Overview

C-THRU is a software library designed to address many of the obstacles encountered with typical networks. It does whatever it takes to establish a network connection in situations where connecting would usually be difficult, and it makes sure that it stays connected with the highest possible quality. It also provides an improved system for addressing applications within a network, permitting true roaming between different connections, and even across different physical types of network.

C-THRU's connection algorithm is able to search for and establish connections faster than existing solutions due to its new method of searching. C-THRU is a small fast library embedded into an application, intended to be imperceptible to end users. It operates over an end-to-end network connection between two applications:



Although its origins are in the VoIP market, C-THRU is equally applicable to many other areas.

C-THRU has been designed with ease of the developer in mind; its features are abstracted through a standard API so as to provide a clean and simple interface which a developer can easily drop into their existing application framework; only a few lines of code need be changed within the existing application and in that respect it is a “drop in” replacement for the existing network interfaces.

Application Interface

C-THRU is intended as a drop in replacement for the well-known Berkeley Socket API; it provides an enhanced set of features though the standard socket interface as well as providing new features which a developer may optionally choose to use.

Integration of C-THRU is intended to be a minimal overhead for developers. C-THRU sits between the developer's application and the underlying OS, transparently catching calls to the socket API.

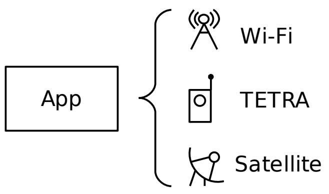
Reduced Development Time

Some of the customers who have been talking with Bubblephone have started simplifying the design of their architecture due to the way C-THRU eases networking and abstracts away complexity leaving system architects free to focus on their applications.

In some instances this simplification represents a reduction of about 40% in the total workload they would have had to undertake on their development.

Multiple Network Types

C-THRU works hard to find the best way to get data where it needs to go, no matter what the circumstances are. This is especially true if multiple uplinks are available; C-THRU's connection engine allows it to search for, establish and maintain connections on many different types of network, not just those using IP.



Versions of C-THRU can be developed for bespoke and unusual types of network if so required, including digital radio systems such as TETRA.

Non-public networks (such as proprietary systems developed for the emergency services) often use custom-designed protocols. C-THRU can be extended as required to communicate over these networks, as per the needs of each application. These networks would not be accessible to other users of C-THRU.

BUBBLEPHONE™

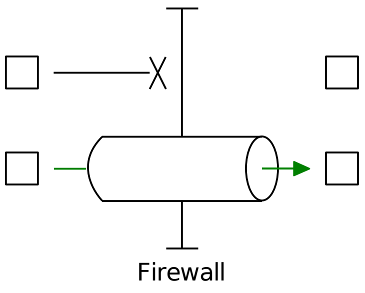
Bubblephone Limited
Sussex Innovation Centre
Science Park Square
University of Sussex
Brighton BN1 9SB
Tel +44 (0) 1273 704 535
http://www.bubblephone.com/

Connecting in Difficult Situations

There are typically many impediments between nodes wishing to exchange data on a network; some are intentional (such as firewalls), some a result of misconfiguration, and some are side-effects of the systems involved (such as NAT preventing incoming connections).

Traditional NAT and firewall traversal techniques address these in a simple sequential approach to selecting connections, without any consideration to change once a connection is established. So they'll try one approach and if that doesn't work, try the next, and the next, and so on. If they do eventually find something that works, they stick with it, even if the situation changes.

In contrast to this, C-THRU's unique approach selects which type is the best connection to make according to the network resources available at any given time. The decision is made by analysing what the network offers: if a connection can be made, C-THRU will find an optimal solution.



Example

A strict firewall will not allow common NAT traversal through, whereas a C-THRU application establishes a connection by some other means, such as tunnelling over acceptable protocols.

This selection may be performed against any quantifiable criteria; typically this is bandwidth (to select the fastest connection). For systems with multiple network types, cost may also be an issue; if required, C-THRU can select so as to automatically use the cheapest network connection where available.

The search is ongoing; if the situation between two points changes, the current solution may no longer be optimal. C-THRU will seamlessly switch connections to another more suitable type. This keeps the Application's connection open and at maximum quality in contrast to traditional NAT and firewall solutions where the connection would typically be dropped.

This allows C-THRU to traverse NAT and firewalls more effectively than existing solutions, and also to deal with with other obstacles; a feature not provided by NAT traversal libraries.

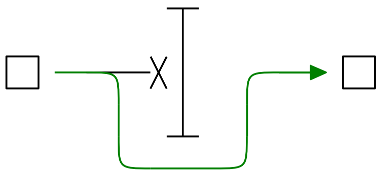
Keeping Connected

Once connected, C-THRU maintains an ongoing awareness of resources available through the network. So it can keep a connection alive whilst changes occur within the network that would cause traditional connections to drop.

C-THRU achieves this by switching to different types of connections when the situational specifics of a connection change. This also includes switching network types if necessary, keeping the socket continuously open during the exchange with no need for the application to reconnect and no loss of data.

Example

If a connection is initiated and a firewall updates with new rules to disallow the current type of connection, C-THRU will detect this and dynamically adjust to keep the connection alive via other means.



This is all done without interruption to the application.

For cleanliness and simplicity this is neatly abstracted away, so applications are only aware of is the same socket API over which they normally send data. Hence C-THRU can maintain open connections where other solutions require the application to reconnect broken network connections.

C-THRU is able to keep these network sockets open in virtually any network weather and at maximum quality for the data transferred.

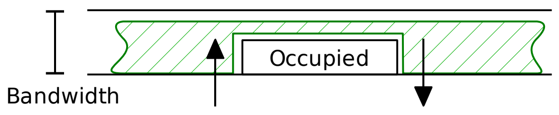
Scalable Compression

As an extension to the Berkeley Socket API, C-THRU can provide developers with optional feedback about what type of compression would best suit the current network conditions. This is particularly suitable for lossy compressions such as codecs for voice and video, where the quality can scale with the available resources.

Compression is often scaled according to the bandwidth available, but may be any criteria defined by the application, such as the financial cost of using a particular network link.

Example

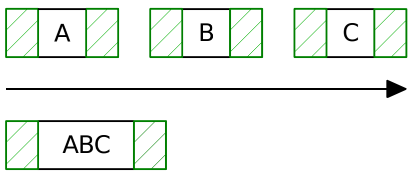
VoIP calls on office networks share their bandwidth with other applications. If a large download starts shortly after a call begins, C-THRU is able to inform the VoIP application to increase its compression.



This allows the audio to fit into the smaller available space rather than attempting to use the same size, which typically causes stuttering and break-up.

This is more flexible than existing approaches which operate with fixed compression levels throughout the lifetime of a connection, helping to reduce problems such as break-up and drop-out on phone calls, video and other time-critical data.

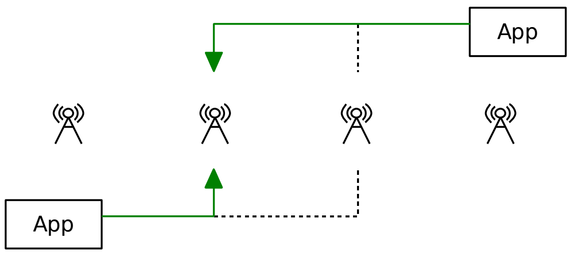
In some situations, C-THRU can reduce latency and the total amount of data needed to be sent during the lifetime of a network connection. By buffering together several separate data packets, it achieves this by reducing the amount of resend requests and carefully selecting an appropriate transfer method to suit that particular type of traffic. The end result is that data arrives faster and with less congestion, making better use of the space available.



True Roaming

Within a user's network there can typically be many layers of NAT causing complexity when connecting to other users. For example an application may have a private IP address, but is also being NATed internally to a non-routable IP. So applications may need to be aware of several addresses, and this gets worse if those address are dynamically assigned, as is usually the case for wireless connections.

With ease of the developer in mind, C-THRU takes care of this complexity by providing a single unique address that the developer uses to identify each client on the network. This abstracts away the complexity introduced by NAT by providing a single address per client.



As a result of C-THRU's system of addressing, true roaming is now possible: a device with a connection established over one wireless access point is able to move and seamlessly switch to another access point without interrupting the application's flow of data.

This works in combination with C-THRU's ability to rapidly switch over to a different connection without the application needing to reconnect to the network.

A technical datasheet is available on request.