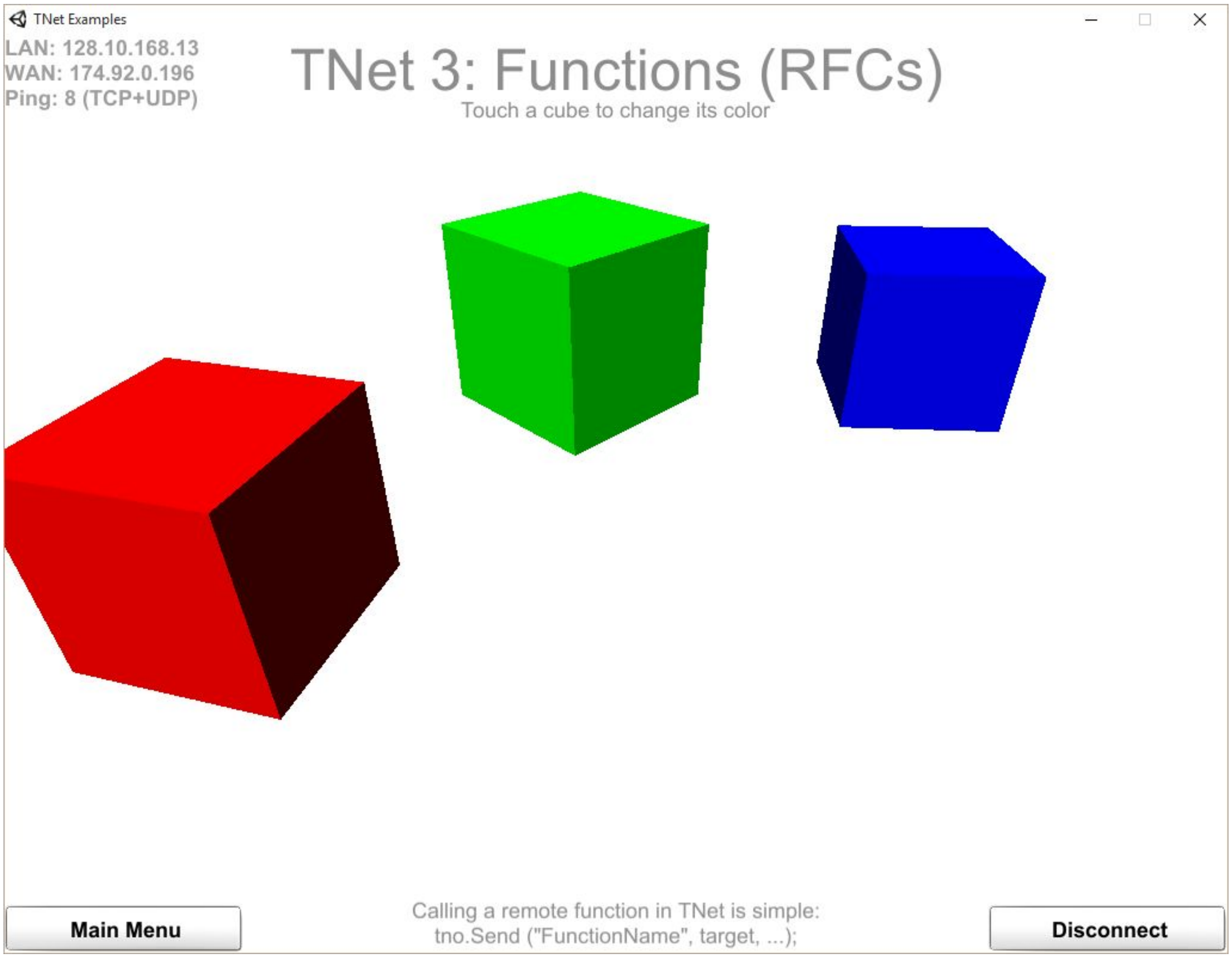
TNet 3: RFC Example Scene In-Depth



LAN: 128.10.168.13
WAN: 174.92.0.196
Ping: 8 (TCP+UDP)

TNet 3: Functions (RFCs)

Touch a cube to change its color

Calling a remote function in TNet is simple:
tno.Send ("FunctionName", target, ...);

Remote Function Call's or RFC's are used to call methods on every remote client attached to the server. They're created by writing a normal method and pre-pending its definition with a [RFC] attribute.

```
[RFC]
void OnColor (Color c)
{ }
```

The RFC Example has three cubes sitting in a scene and when clicked, they will change their colour and transmit the colour change over the network to any other client viewing the scene.

It does this using three C# scripts. The first TouchHandler script is attached to the Main Camera. It's a helper class which detects mouse clicks and then broadcasts them out as simple function calls such as `OnClick` and `OnPress`.

Each cube in the scene contains two component scripts. The first is the `TNObject` "Network Object" script. Every object which will interact over the network needs to have this script attached. It does a number of operations behind the scenes, but outwardly it assigns objects a unique ID number.

If your object containing the `TNObject` is spawned by TNet, the ID will be assigned automatically and you don't have to worry about it. However, if you are placing the object in the scene manually, as it has been in this example, the number needs to be set by hand. In this case, their ID's are 1, 2, and 3.

Finally, the most important script is the `ColoredObject` script. The class inherits from `TNBehaviour` as opposed to `MonoBehaviour`. That gives the script access to `tno` - a variable which accesses the TNObject properties such as `.owner`.

```
public class ColoredObject : TNBehaviour
```

A little farther down you'll see the OnColor event with the [RFC] attribute in front of it.

```
[RFC] void OnColor (Color c) { mMat.color = c; }
```

This function is setup to receive a `Color` over the network and assigns it to the previously cached material.

```
void OnClick ()
{
	Color color = Color.red;
	if (mMat.color == Color.red) color = Color.green;
	else if (mMat.color == Color.green) color = Color.blue;
	tno.Send("OnColor", Target.AllSaved, color);
}
```

The `OnClick` function is what brings everything together. It responds to the `OnClick` broadcast from the TouchHandler and toggles the colour of the material between red, green, and blue. This only executes on the Client where the user clicked.

`tno.Send()` is what sends the colour change over the network. The RFC “OnColor” on all clients will receive the `color` variable. The parameter `AllSaved` means that first, it will be sent out to all clients connected to the network, including our own. And second, it will be saved on the server in case another client connects. When that new client connects, the RFC will be sent to him automatically before that player finishes joining the channel, making RFCs ideal for preserving active states.

If the channel was marked as persistent by passing ‘true’ along with the first call of TNManager.JoinChannel, the RFCs will also be saved on the server when the server gets shut down and loaded when the server gets started back up -- in effect preserving the state even through the server restart.