# FORGE NETWORKING

# Contents

## Useful Links

Support Email: support@beardedmangames.com

Main developer portal: http://developers.forgearcade.com/

Main tutorial portal: http://developers.forgearcade.com/Tutorials

Main Forums: http://forums.forgearcade.com/

Main Feedback Portal: http://developers.forgearcade.com/Feedback

Main Unity Forum Thread: http://forum.unity3d.com/threads/superpowered-cross-platform-networking-library-including-windows-universal-apps.286900/

Beginner Videos:
https://www.youtube.com/playlist?list=PLaMPnrbpanJ7DazlqTkZHHBSt8D5dx4Um

Intermediate Videos:
https://www.youtube.com/playlist?list=PLaMPnrbpanJ6DC4RmaEbyHgcnqi6HrfTt

# Welcome

Welcome to the initial release of the **Forge Networking** system. We would like to first thank you so much for supporting the system through your purchase. We here at Bearded Man Studios, Inc. are a small indie development team that is passionate about software and games. We are excited to have you on-board and using this networking solution! I want to talk briefly about our networking platform and give a quick snippet of who we are and what we are trying to achieve. At Bearded Man Studios, we are trying to make a networking solution that is multi-platform and that can suit any of the vast networking needs in games. When we initially approached this networking solution, we wanted something that would work with the game designs that we imagined. Seeing that there wasn't anything on the market that could meet those needs or even sustain them without becoming overly costly, we decided to venture into developing our own networking solution. As we dove into making this, we discovered that not only did we think it was something we were proud to work on, but something that we believed that others should be able to use as well. That was when we decided to share this networking system with other dreamers like ourselves.

As indie developers, we are dreamers. We have huge plans and hopes for the things we develop and we are proud of our products! The developers here at Bearded Man Studios have been developing with Unity since its earliest days and we love the engine! Some of us are even working at the second largest Facebook gaming company in the world with a Daily Active Users (DAU) in the millions. However, when we go home, we are indie dreamers working on our indie games and our indie networking solution.

We are deeply interested in every developer that uses our platform. Please feel free to contact us at any time, we are honored to have our system be a part of your dreams (games). Please feel free to socialize with the community and help Forge grow into a powerful force. With time, we hope that Forge will enable all developers to make multiplayer games and applications quickly and easily. Please report any bugs you may find, request any ideas or suggestions that you have and give us your feedback. All of the feedback is essential to developing a system that not only works great for you, but for us as well!

Thank you,

- Brent Farris

President/CEO and Lead Software Architect

## Getting Started

Forge Networking is an all-in-one networking solution for Unity. With Forge Networking, you are able to create networked games and applications extremely fast without any need for a heavy network engineering background. If you do know networking and wish for a system that you can have full control over, Forge Networking comes with complete source code for all platforms out of the box! This document will go over the main concepts of Forge Networking and will introduce you to the system with a step by step example.

## Specifications

| Available Features | Available Platforms |
|---|---|
| - UDP/RUDP | - Windows (x86/x64) |
| - Remote Procedure Calls | - Mac (x86/x64) |
| - Complete Unity native C# source code | - Linux (x86/x64) |
| - Multi-Threaded | - iPhone / iPad |
| - Event Driven | - Android |
| - HTTP/HTTPS libraries | - Windows Phone 8.1 |
| - TCP | - Windows Store 8.1 |
| - Writing of raw data | - Windows Universal |
| - Header sizes ranging from 1-32 bytes | - Webplayer |
| - Data compression | |
| - Data updates on Deltas | |
| - Individual object update intervals | |

## Data Compression, Performance and Bandwidth

Forge Networking comes with delta data compression and bandwidth reduction out of the box. In our modern world, reducing bandwidth and boosting performance whenever possible is key to making our users satisfied.

### Delta Value Compression

Forge Networking tracks the variables that are being serialized across the network and their changes. What this means is that a variable that is not changing will not be updated across the network until its value has changed. So let's say that you have a position to an object in 3D space (Vector3). If the object does not move, the position of that object will not be distributed across the network as that would be a waste of bandwidth due to duplicate data.

However, the moment that the object moves and that position is updated, the position will then be distributed across the network until it is no longer updated.

## Performance

In Forge Networking, we continuously work towards performance anywhere possible. We even have modified versions of native C# classes in order to boost their performance in the system. Our biggest performance boost comes from reducing Garbage Collection within the system. Since bytes are created all the time and in many areas, we introduced the BMSByte which is a caching strategy for longer living bytes. The BMSByte, without it getting too much into the nitty-gritty, is a class which is used for streams that are often used or re-used in order to prevent constant garbage collection.

Forge Networking is completely multi-threaded and pushes long running processes to separate threads. Connections and reading of incoming data are both performed on a separate thread from the main one. By using multi-threading, we can gain an extra boost of performance so that the main thread is not blocked by network threads actions. Our idle CPU performance has been estimated to be lower than every other system that our beta testers have used (per their reports to us). We will work on producing actual numbers and graphs on how our system's performance is, compared to others.

## Bandwidth Reduction

Bandwidth is the amount of data that is being sent across the network. It is very important to reduce bandwidth in a networking system for many reasons. Here are a few:

1) More people use mobile which charges for excessive bandwidth usage
2) You as a content provider may use cloud computing which charges for bandwidth usage
3) Ultimately if less is sent across the network, better response times are achieved

In Forge Networking, we reduce bandwidth greatly by minimizing the headers of each of our packets. Packet header sizes found in other popular networking systems are huge, clunky and can cost developers copious amounts of money and response time for no reason. In Forge, everything including RPCs (Remote Procedure Calls) is mapped to smaller data types. By mapping data types, we reduce unnecessary bytes and can achieve a very low header cost. The maximum header size that Forge Networking produces is 32 bytes, and the minimum header size in the system is 1 byte (4 times smaller than when you create an "int" variable on 32bit systems).

## Questions & Answers

***Q: Do I need to know network programming to use Forge Networking?***

A: No, though to do complicated behaviors across the network you will need to have some level of programming knowledge.

***Q: Do you suggest any tutorial or guide on how to do network programming or at the very least how to use Forge Networking?***

A: Yes! We are actively developing networking tutorials to not only teach you how to use Forge Networking, but also to actually introduce some knowledge in network engineering itself. You can find these tutorials here: http://developers.forgearcade.com/Tutorials

***Q: What header sizes in bytes can I expect to be generated in the system?***

A: Our internal header sizes range from 1 byte to 32 bytes during standard communication.

***Q: Can I control how my data is being sent across the network rather than relying on the default method?***

A: Yes! You can use our built in WriteRaw method which produces a header size of 1 byte to serialize data however you wish.

***Q: How does this compare to other networking systems and the upcoming UNET system?***

A: There are many advantages for using Forge Networking, such as direct developer support, full source code, active networking, exclusive community, tutorials, Unity native C# code, cross platform compatibility, support for TCP networking, multi-threading, custom communication protocols, and much MUCH more!

***Q: Do the developers of Forge Networking actually use the system?***

A: Yes!! We are actively developing our own games using the system! Come, join us!

***Q: Can I make tutorials using Forge Networking?***

A: Yes! But please refrain from showing any of the source code for Forge Networking in your videos. We trust every developer with the open source code to be responsible with it! ☺

# Step By Step Example

## Before You Start

1) No programming experience is needed to follow this tutorial
2) You need to have Unity version 4.6 or higher installed
3) You will need to have the Forge Networking package

## Importing the Package

To import our package into Unity, just do one of the following:

1. Drag it from your file system into the unity window and drop it onto the "*Assets*" folder
2. With Unity open, double click on the package and then click on *import* when prompted in Unity
3. Locate your purchase in the *Unity Asset Store* and click the "*Import*" link. If you have not yet downloaded it, you will need to click the "*Download*" link before the "*Import*" link becomes available

## Create a Scene

To create a new scene, just go to *File* -> *New Scene* or use the appropriate hot-keys. Once this is complete, you should save your scene to the file system. Click *File* -> *Save Scene* and when prompted, save it to your Unity project named "*Game*".

## Making a Networked Cube

In your new scene click on GameObject->3D Object->Cube. You will see a new cube in your scene Select the cube and then set its position to (0, 0, 0) so that it is visible by the camera. Now click on the "Add Component" button in the "Inspector" of Unity while the cube is still selected. Type in "NetworkedMonoBehavior" and you will see a C# script that is named that. Select that script to add it to the object.

## Making the Cube Move Across the Network

To see that the cube is being networked, we will add a script to it that makes it move. Right click on the "Assets" folder and then go to *Create* -> *C# Script*. Name this script "*DemoMove*" and then open the file to be edited. Make the script look like the following and then save it:

```
using UnityEngine;

public class DemoMove : NetworkedMonoBehavior
{
        Protected override Update()
       {
               Base.Update();

               If (!IsOwner)
                       Return;

               if (Input.GetKeyDown(KeyCode.UpArrow))
                       transform.position += Vector3.up;

               if (Input.GetKeyDown(KeyCode.DownArrow))
                       transform.position += Vector3.down;

               if (Input.GetKeyDown(KeyCode.RightArrow))
                       transform.Rotate(Vector3.up, 45.0f);

               if (Input.GetKeyDown(KeyCode.LeftArrow))
                       transform.Rotate(Vector3.right, 45.0f);
       }
}
```

## Preparing Your Build

Now that you have altered the scene, make sure to save it with *File -> Save Scene*. Next, you need to open up the Build Settings by clicking *File -> Build Settings*

First, we are going to make sure that we have "*Run In Background*" turned on for our build so that the network communications continue while the game is not focused on our desktop. Select the "*Player Settings*¦" button located in the lower left corner of the build settings window. Open the settings section labeled "*Resolution and Presentation*" in the inspector window. Now just click the checkbox labeled "*Run In Background*" so that it is turned on.

Next, while the *Build Settings* window is still open, navigate to the *Bearded Man Studios, Inc. -> Networking-> Scenes* folder and drag the "*ForgeQuickStart*" scene into the build settings area labeled "*Scenes in Build*". Next, navigate to your "*Game*" scene and drag it into the same area but below the "*ForgeQuickStart*" scene.

Under the "*Platform*" area of the Build Settings, click on "*PC, Mac & Linux Standalone*" then click the "*Switch Platform*" button on the lower left.

## Running the Build

Now click "*Build and Run*" and then save the game onto your hard drive. Once the game opens (accept any firewall permissions you are prompted with), click on "*Start Server*", then go to the Unity Editor and open the "*ForgeQuickStart*" by double clicking it. Press play in the editor and then click "*Start Client*". Now you can click on your running PC build and press any of the arrow keys.

## Congratulations

You now have glorious networking!

## Included Example Scenes

The Forge Networking comes with an array of example scenes to show various parts of the system working. Here is a quick overview of each of the scenes, their purpose, and the information they are trying to display.

**ForgeQuickStartMenu**

This is a menu scene that can be used to start a server, locate a server on the local area network, join a server by IP address or go to the default server browser.

**ClientTimeout**

This scene demonstrates how client timeouts work and how to handle them.

**ErrorHandling**

This scene shows how to handle errors that are thrown in Forge Networking so that you are able to sort out issues at runtime.

**ForgeAutostartHeadlessLinux**

This scene shows you how to automatically fire up a server instance. Preferably, this would be a headless Linux or just a Unity instance in "batch mode". See Unity's "*Command Line Reference*" page for more information: http://docs.unity3d.com/Manual/CommandLineArguments.html

**ForgeHelloCube**

This is a bare-bone scene that has a cube that moves around. This cube is instantiated by reference and not by resources.

**ForgeHelloCubeResources**

This is a bare-bone scene that has a cube that can move around. This cube is instantiated from the resources directory unlike the "*ForgeHelloCube*" example scene.

**ForgePureRPC**

This scene is used to demonstrate how “*Pure RPC*” can be achieved with Forge Networking. *Pure RPC* is the concept that objects are not constantly serialized across the network by their updates but instead by information that is sent with only Remote Procedure Calls.

**ForgeQuickServerBrowser**

This is a simple server browser scene that can be used in conjunction with the *ArbiterAPI* in order to create server listings.  Whenever a server is started and the API key has been assigned, servers will be registered to the global database much like how Unity has host listings.

**ForgeWriteCustom**

In this demo scene we go over how to write our own custom classes across the network.  This will allow the developer to serialize entire custom classes and custom data across the network in order to communicate in a more complex way.

**ForgeWriteRaw**

This scene goes over how to use the built-in “*WriteRaw*” method.  This is a method that has a 1-byte header (the smallest header allowed) and allows developers to write custom data across the network however they wish.  This method will allow developers to create their own custom messages to be sent in order to optimize data compression to a maximum.

## More Information

For more information on how to use the system, please visit our tutorials page on the developer's website (http://developers.forgearcade.com/Tutorials). These tutorials are constantly being developed and modified, so we will be unable to show all of them in this document. We are currently working on converting all of our video tutorials to text-based tutorials as well in order to extend the support to our non-English speaking friends!

# Recent Version Changelists

## Release V1:

- Moved all code from DLL to Unity
- Fixed buffered RPCs not clearing correctly
- Fixed issue with RealSenderId constantly being set to server identity
- Using NETFX_CORE preprocessor so Windows Store and Windows Phone do not need a plugin DLL
- Fixed connection issues with connecting Windows Store applications to servers on other operating systems
- Added new MainThreadManager class, just call MainThreadManager.Run(method); to run a method on the main thread
- Removed the duplication of main thread action handlers from NetWorkerKiller and ArbiterAPI
- Update HTTP to have better support for get parameters
- Improved stability of threaded HTTP request
- Added HTTPS request fixes
- Fixed Arbiter API to register online servers correctly
- Fixed server browser

## Beta V14 Changelist:

- Fixed TCP so that it is now working along with the new UDP standards

## Beta V13.6 Changelist:

- Fixed server reading raw write bytes 2 times
- Fixed issue with byte offset on raw read being 2 and not 1, ***** If anyone has their code to skip the first two bytes, this is not necessary as the extra 0 is no longer after the 1 (first byte).
- Now using heavily modified phpBB for developer portal forums

## Beta V13.5 Changelist:

- Fixed issue with NetworkingManager causing errors with no network
- Added an initialization call to NetworkingManager once a connection is established
- Fixed issue with Unity 5 having null refs on normal built in references

**Developer Portal**

- Prevent setting username as email
- Show the bug/feature reporter username (if not email address)
- Show the username of the person who replied on bugs/features
- Do not allow replies from users who have not finished setting up their accounts on bugs/features
- Fixed issue with forum threads containing special characters
- Fixed issues with forums not opening specific links

## Beta V13.4 Changelist:

- Fixed race condition on hosting/connecting and events

**Bare Metal**

- Fixed non-connecting issues as they are related to CallOnMainThread issues

## Beta V13.3 Changelist:

- Fixed initialization issue with arbiter api
- Updated Exception logic
- Can now throw an exception event with NetWorker.ThrowException
- New exception handling sample scene
- The NetWorker error event now just takes in any exception

## Beta V13.2 Changelist:

- Re-connecting to the same port gets an error unless the application is fully closed
- Make the server call the read raw event as it is sending it off to all of the clients

- Client Timeout event added - If a client doesn't get a response from the server in the specified milliseconds it will timeout and fire the timeoutDisconnected event

**Developer Portal**

- Thread following
- Forum notifications for followed threads
- Fixed feedback not filtering complete features
- Fixed invalid access on posing threads

## Beta V13.1 Changelist:

- Fixed collision issues when spawning objects where there was "floating" collision

**Developer Portal**

- Forum threads can now be followed
- Forum notifications (on site in forums) for followed threads on posts

## Beta V13 Changelist:

- Networking class now can ping a host and get the response time
- Fix issue with disconnect being called at start
- Client hangs and doesn't close, the instance needs to be killed
- Updated Win8.1 and WP8.1 libs

**Developer Portal**

- Require account to finish setup before posting on forums
- Prevent locked threads from showing input box
- Developers can now respond to each other’s bug reports and feature requests
- Updated parent database object to do correct valid checks on blank new objects