A (very) brief guide to networking with XNA

To start with, I recommend downloading and skimming through the peer-to-peer sample from the Creators Club website. This sample can be found at: http://creators.xna.com/en-US/sample/networkp2p

In the constructor for the PeerToPeerGame class (default would be Game1 or a new XNA project) there is an important line of code needed to get your game networked:

```
Components.Add(new GamerServicesComponent(this));
```

The XNA Game class has a list of GameComponents. A GameComponent can be thought of as a gameobject that takes care of itself. That is, once you add it to the list (e.g. Components.Add(myComponent)), the component will update and draw itself on its own. This means you don't have to worry about managing your component after you've created it and thrown it into your game's list. The GamerServicesComponent wraps the GamerServicesDispatcher which manages the signing in of profiles, and the displaying of the Guide (Xbox Live or Games For Windows Live Guide). We need both of these things for our networking.

Once we have our GamerServicesComponent, we're ready to do the networking. The peer-to-peer sample gives us a few important methods to do just that:

UpdateMenuScreen(), CreateSession(), JoinSession(), HookSessionEvents(), GamerJoinedEventHandler(),

SessionEndedEventHandler(), UpdateNetworkSession(), UpdateLocalGamer(), and ReadIncomingPackets().

All of these methods can be copied directly from the sample and be used in your game. If you look at the code, these methods are fairly easy to follow in terms of what they do, which involves setting up and keeping track of the network connections. This is code you can reuse in any game you want to make locally networked. The last two methods in the above list (UpdateLocalGamer(), and ReadIncomingPackets()) are the ones you want to take a closer look at. Here is where you can send your game data between computers or Xboxes. We use the LocalNetworkGamer.sendData() and receiveData() methods to send and receive packets respectively. The real meat of these methods is where the packetWriter and packetReader objects are used. If you look at the last line of the UpdateLocalGamer() method :`gamer.SendData(packetWriter, SendDataOptions.InOrder);`, you see SendDataOptions.InOrder. This ensures that newer packets never arrive earlier than older packets. From here, all you have to do is write to your packetWriter any objects you want to send off (these include any types supported by the C# BinaryReader/Writer classes as well as common XNA types

such as Matrix and Vector2/3/4) before the call to sendData(). Then in the receive data method you can read the same objects in the same order from the packetReader and interpret them as you see fit. For example, the position of a game object can be sent from computer to computer using a Vector3. Since both instances of the game know about the object and how to draw it, they don't need to send the entire object every time, just the necessary data so that the user is seeing an accurate representation of the model (in this case, it might be the location of where to draw the hero).

That's pretty much all there is to it. Good luck and happy networking!