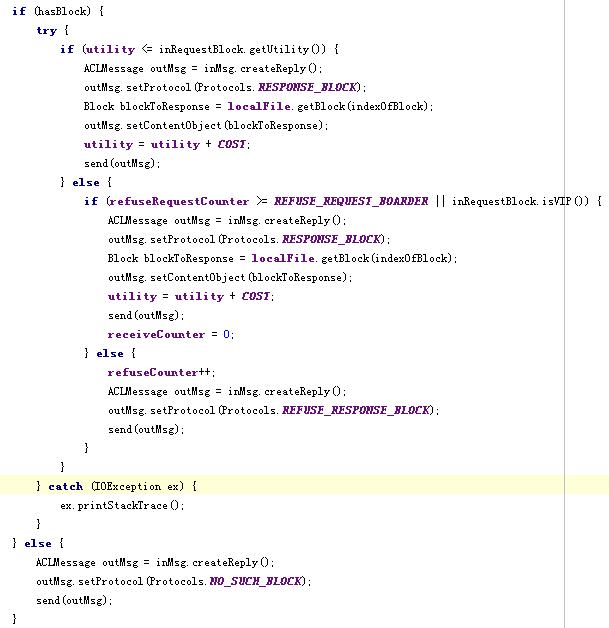
Establish Nash Equilibrium in File Sharing Application

In our file sharing application, for each peer we mainly focus on the following two parameters, cost and benefit, and the utility of this agent calculates as cost minus benefit. Initially, when a peer joins to the file sharing, its cost and benefit are both zero. After the peer downloads a block, its (obtained) benefit increases by one, on the other hand, the peer’s cost raises up one unit provided that it’s uploading/sharing a single block.

Each and every peer maintains its individual utility whenever downloading or uploading a file block. When one peer wants a block from another peer, it piggybacks its current utility with the request of the block number. After received a request, a peer checks its current utility against the one (from requesting peer) embedded within the request, if the other utility is equal or greater than its utility, the request is granted, otherwise a global refuse counter is inspected, if the refuse counter is larger than the predefined max refuse number, the request will be however granted, in case the refuse counter is still small, the request will be refused, as well as the counter increases by one.

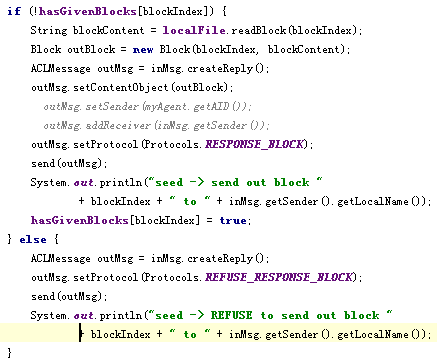


Figure

C:\Documents and Settings\Tom\Desktop\Drawing1.tif

Figure

In respect of the seed agent, we codes it in the way, it will send out each and every file block once and only once, to ensure that the other peers can eventually retrieve the whole file by asking each other rather than seed, so the seed agent will refuse the further requests of blocks from any client agent. The following code shows the implementation.



Figure

The text file “**output.txt**” illustrates the result of our file sharing application. One can see that every client asks different block of file from different peer, and their utilities are quite similar to each other, which means all peers are actively participating the file sharing, not only downing but also uploading. No free-rider exists in our application and the seed is released from the heavy request from other clients.