101 Solutions

Iteration 2

101 Solutions

August 18, 2013

Contents

1 Introduction

The purpose of this document is to keep track of the ongoing development progress of the software application manager AppMan. In each iteration additional content will be added which will result in this document being an up to date and relaible method to view progress and current development plans and ideas. Content will include milestones reached, current goals and the current functionality of the software.

2 Milestones reached

- A cleaner GUI implementation
- Server running on the Master - AppMan
- Client connection to the master through network
- A reliable socket connection between the server and client applications, which will serve as a communication pathway
- Build add to the master computer
- View builds on the master
- Creating a database that can store information related to builds kept on the master and slaves

3 Immediate Goals

Our next main goals for the project:

3.1 Build transference

- Goal: The ability to copy the builds to the clients(Slave computers) and also to compress the files that will be copied

3.2 Build Information Comparison

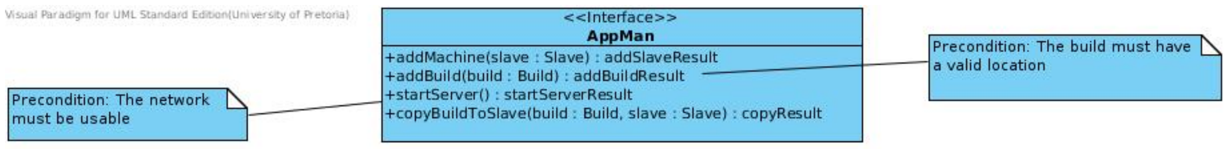
- Goal: The ability to synchronize two builds on master and slave computers(i.e. if a change have been made to a file)

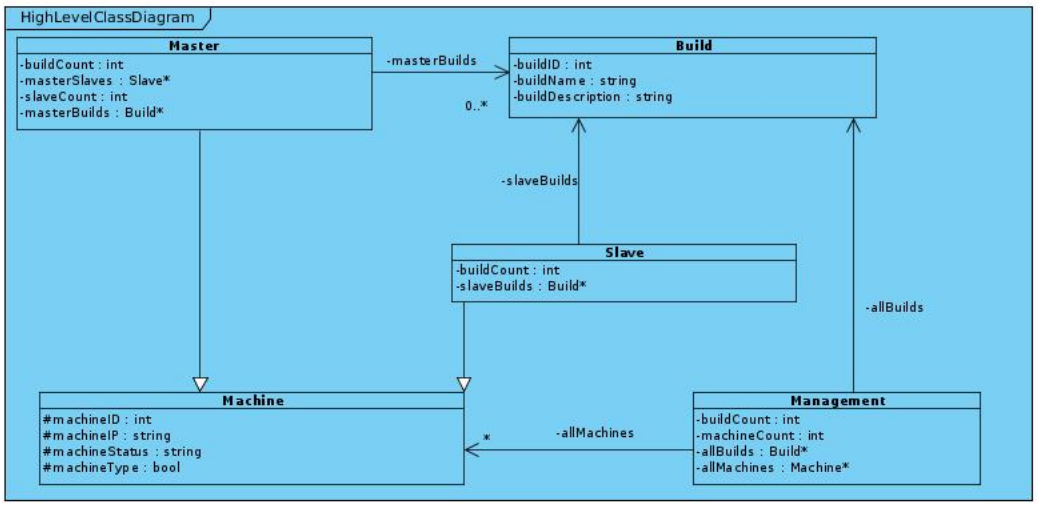
3.3 Slave Build Display

- Goal: The ability to show the builds on slave machine

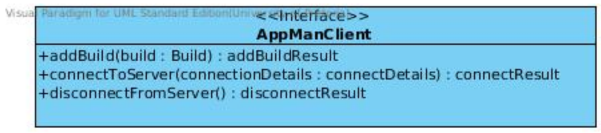
4 System Design

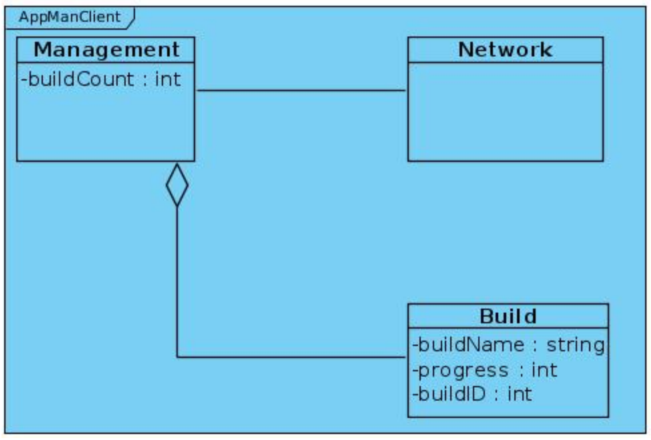
4.1 AppMan


Visual Paradigm for UML Standard Edition(University of Pretoria)
<<Interface>>
AppMan
+addMachine(slave : Slave) : addSlaveResult
+addBuild(build : Build) : addBuildResult
+startServer() : startServerResult
+copyBuildToSlave(build : Build, slave : Slave) : copyResult
Precondition: The network must be usable
Precondition: The build must have a valid location



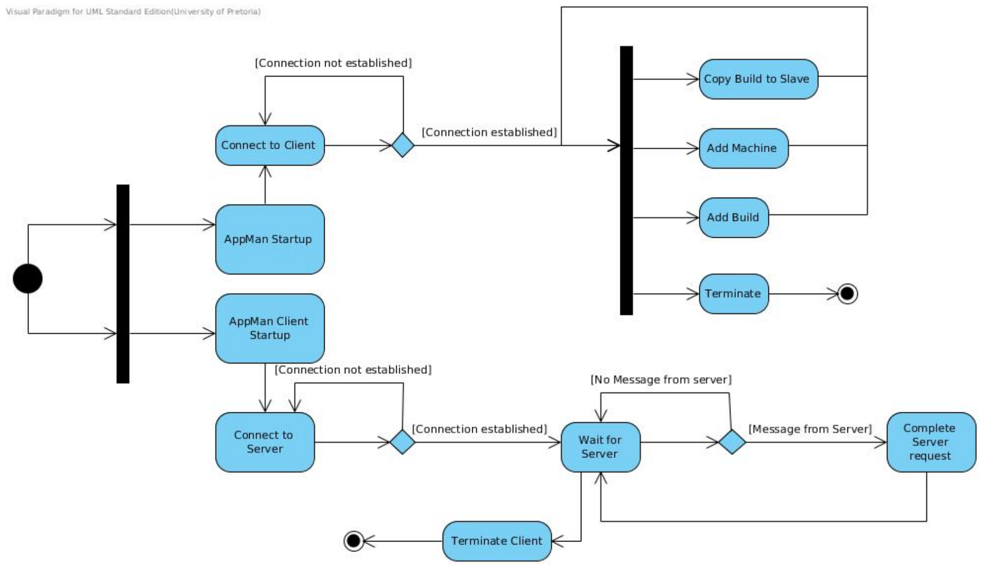
4.2 AppManClient


Visual Paradigm for UML Standard Edition(University of Pretoria)
<<Interface>>
AppManClient
+addBuild(build : Build) : addBuildResult
+connectToServer(connectionDetails : connectDetails) : connectResult
+disconnectFromServer() : disconnectResult



5 Processes

5.1 ActivityDiagram

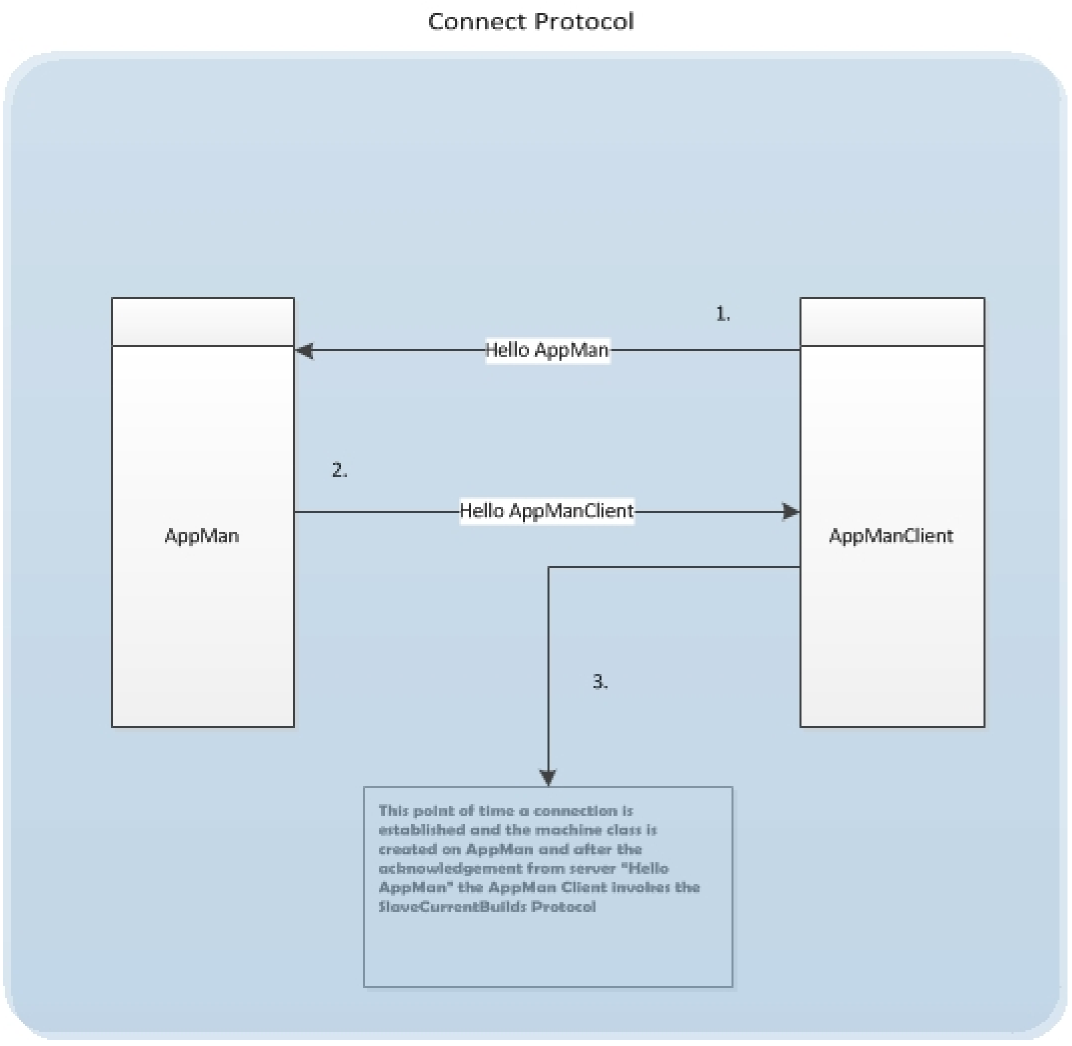


6 Communication Protocol

Brief Description: The following diagrams are Data Flow Diagrams describing the protocol that is used to implement communication between the AppMan and AppManClient. The protocol is easy to understand and each of them serve a purpose in order to harbour proper communication. The order in which the communication takes place are indicated by the numbers near the flow.

6.1 Connect

This protocol is followed when the client application first connects to the AppMan server. The connecting application will directly be disconnected if the correct protocol is not followed. SlaveCurrentBuilds protocol directly follows this protocol.

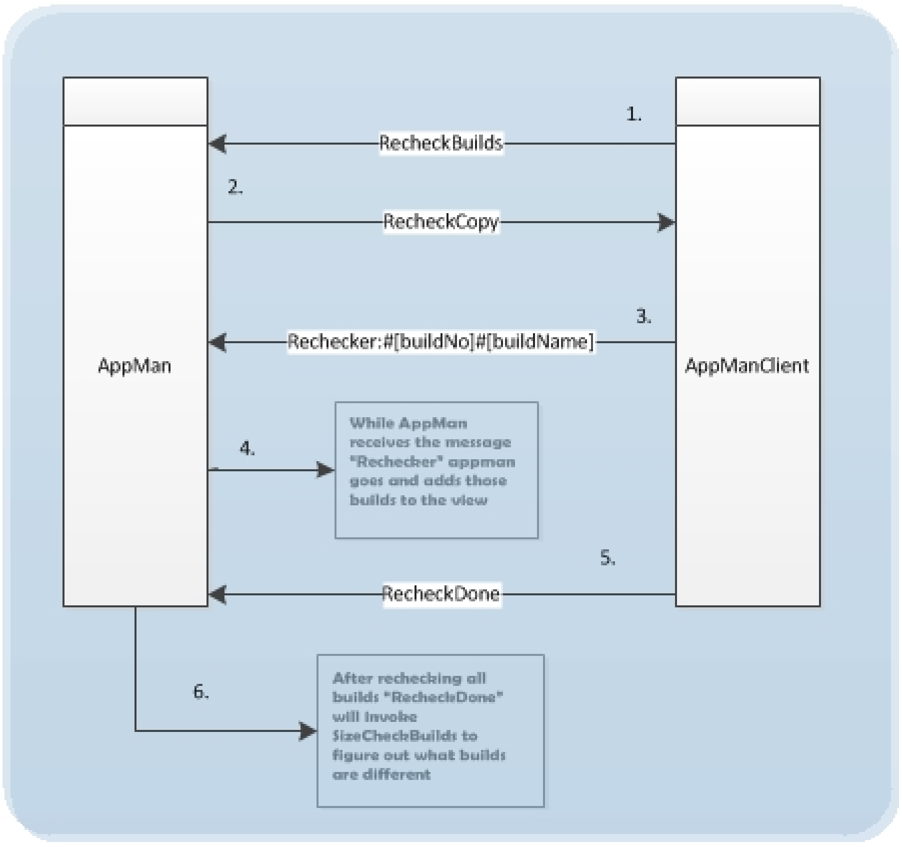


This point of time a connection is established and the machine class is created on AppMan and after the acknowledgement from server "Hello AppMan" the AppMan Client involves the SlaveCurrentBuilds Protocol

6.2 SlaveCurrentBuilds

This protocol is invoked directly after Connect protocol and will update all the builds that are currently on the slave and show them on the master. With this protocol one can see all the builds that are there. Apon reaching the "RecheckDone" data, the SizeCheckBuilds protocol will be invoked for each build.

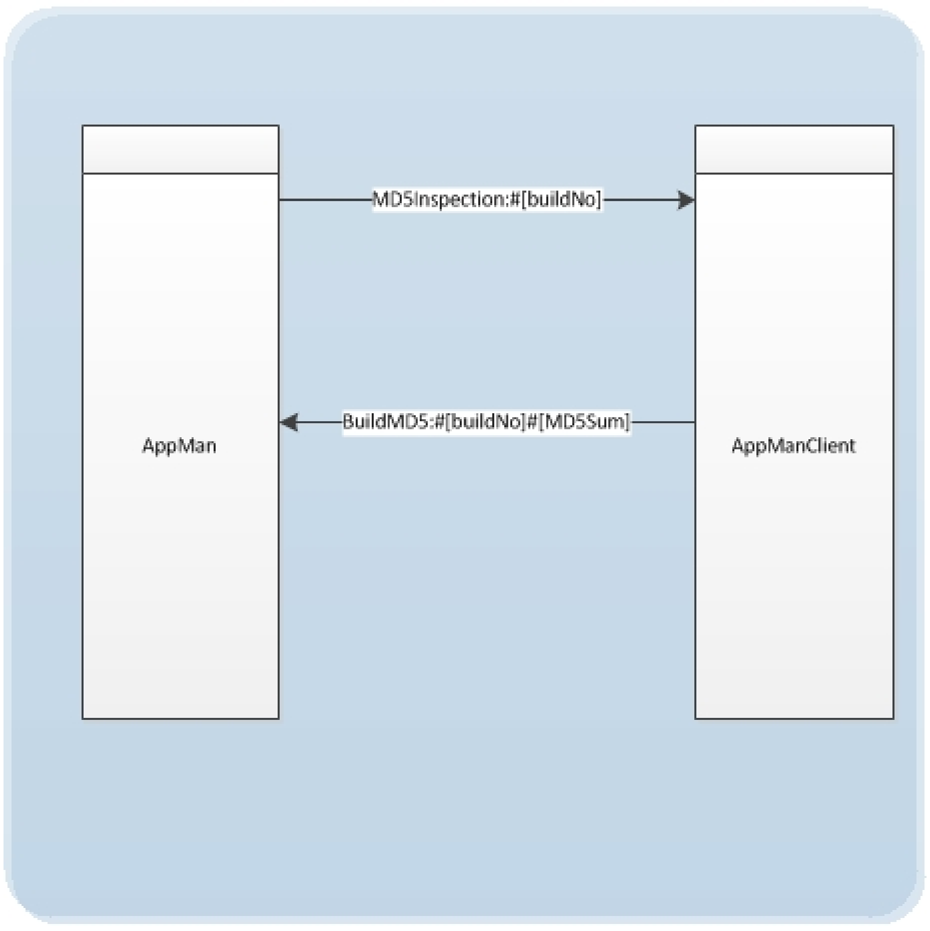
SlaveCurrentBuilds Protocol


AppMan
AppManClient
1.
RecheckBuilds
2.
RecheckCopy
3.
Rechecker:#[buildNo]#[buildName]
4.
While AppMan receives the message "Rechecker" appman goes and adds those builds to the view
5.
RecheckDone
6.
After rechecking all builds "RecheckDone" will invoke SizeCheckBuilds to figure out what builds are different

6.3 SizeCheckBuilds

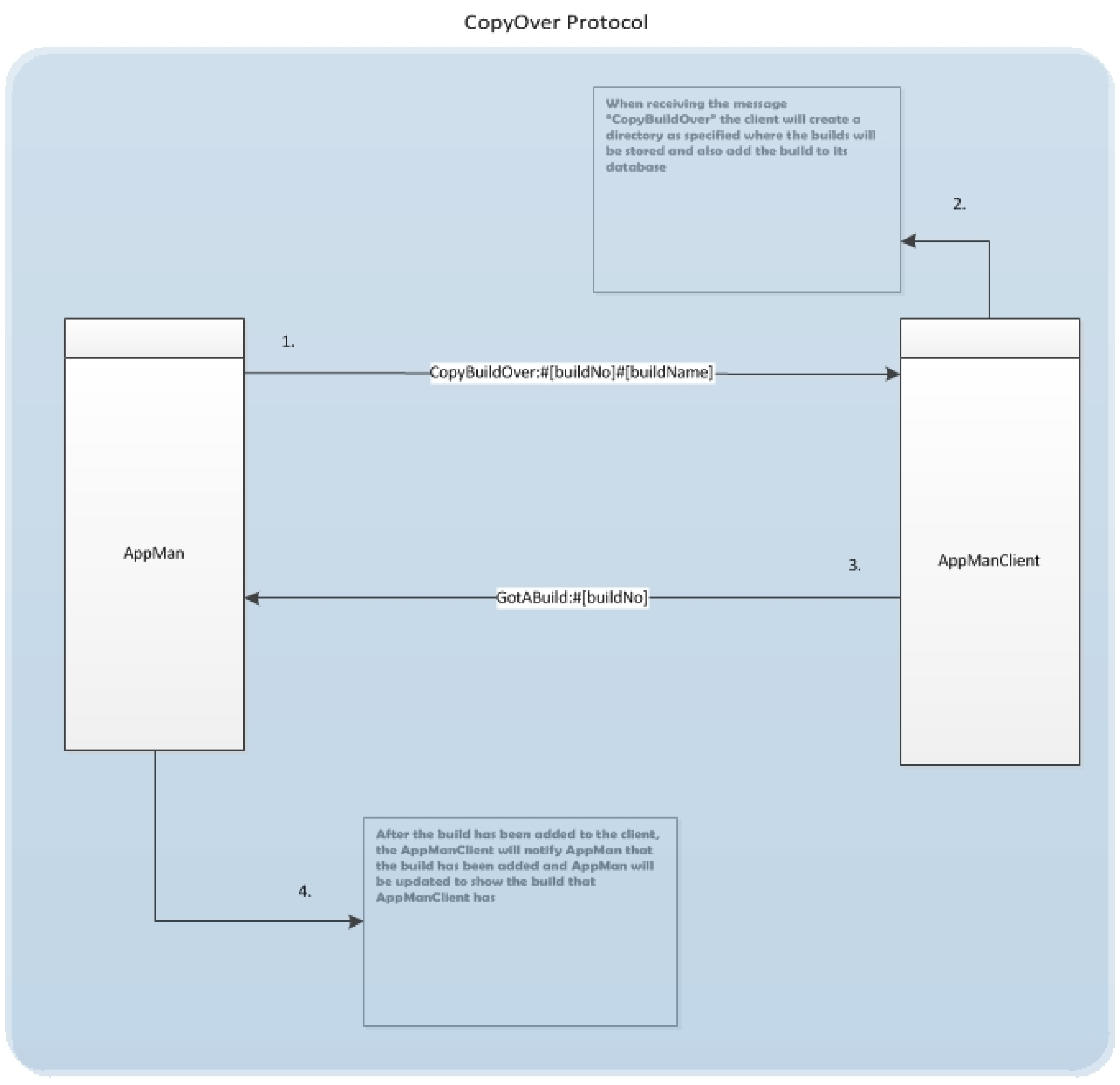
This protocol is followed when determining if the build of the AppMan and AppManClient have different MD5 values which will indicate that there are differences in the builds themself. If a difference is detected CopyOver protocol is invoked to copy the build files over from the Master machine to Slave Machine.

SizeCheckBuilds Protocol



6.4 CopyOver

This protocol is followed when a build is logically copied over from master to slave. The build name and number is saved on the slave side.

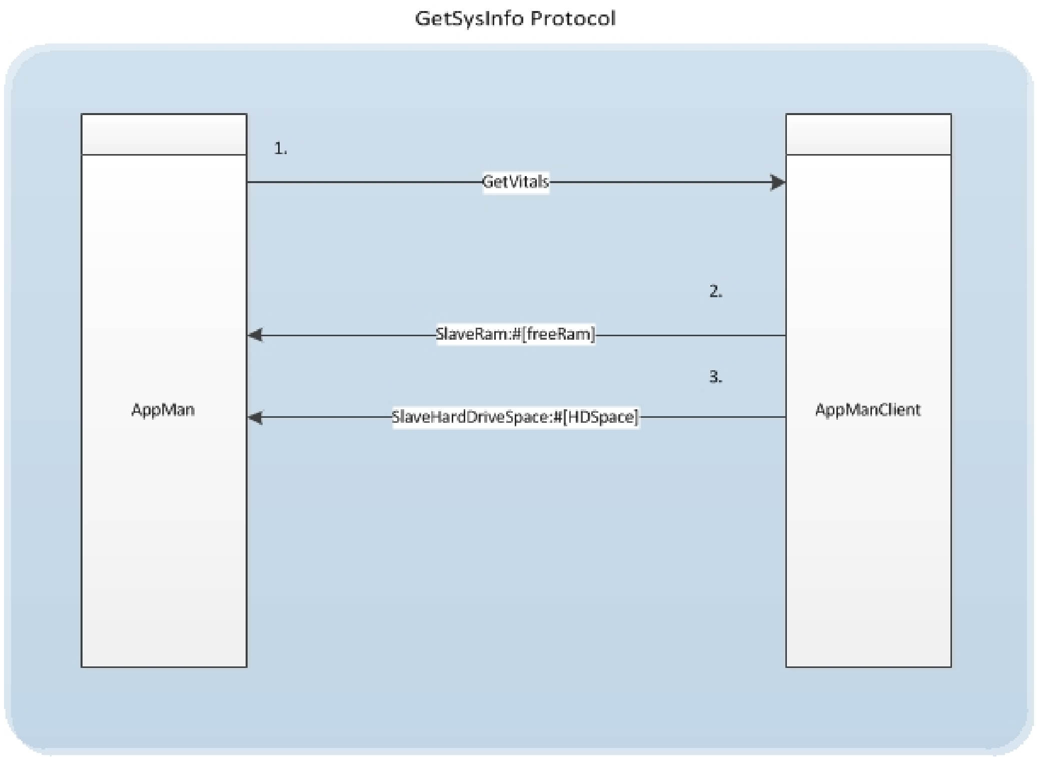


When receiving the message "CopyBuildOver" the client will create a directory as specified where the builds will be stored and also add the build to its database

After the build has been added to the client, the AppManClient will notify AppMan that the build has been added and AppMan will be updated to show the build that AppManClient has

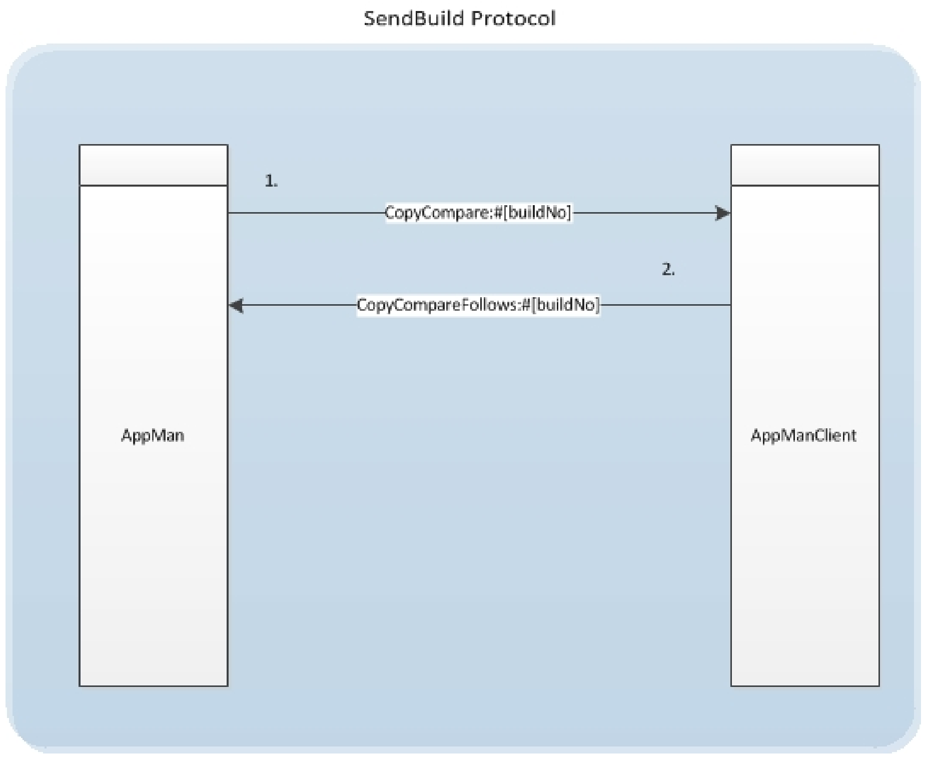
6.5 GetSysInfo

This protocol is used to obtain various system info from the slave machine to display on the master machine. The slave machine returns its ”vitality” which is displayed on the master machine.



6.6 SendBuild

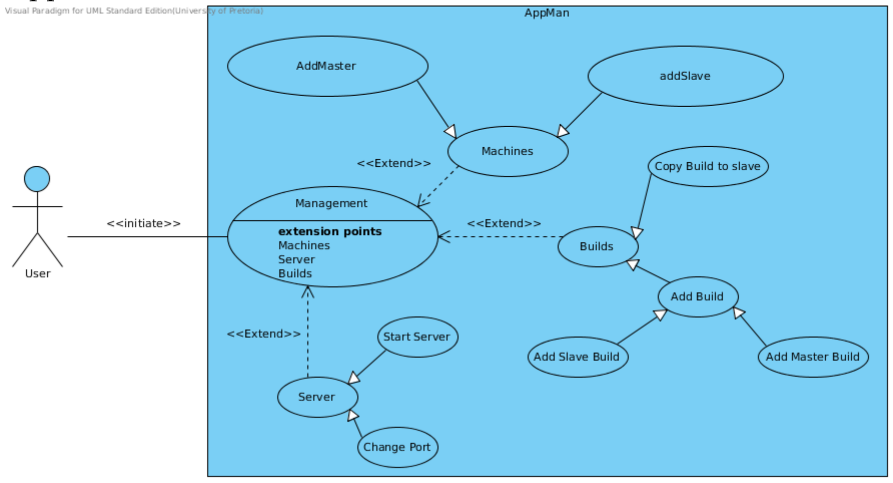
This protocol is where a physical build is being sent to the slave machine. Before sending a build, the MD5 values are calculated for each and every file so that one can see what files are different. The MD5 values sent to the master will be used to create a CopyCompare class which will indicate which files will be sent across the network. The file that is sent across the network will be uncompressed and those files will be added in an uncompressed format on the AppManClient side.



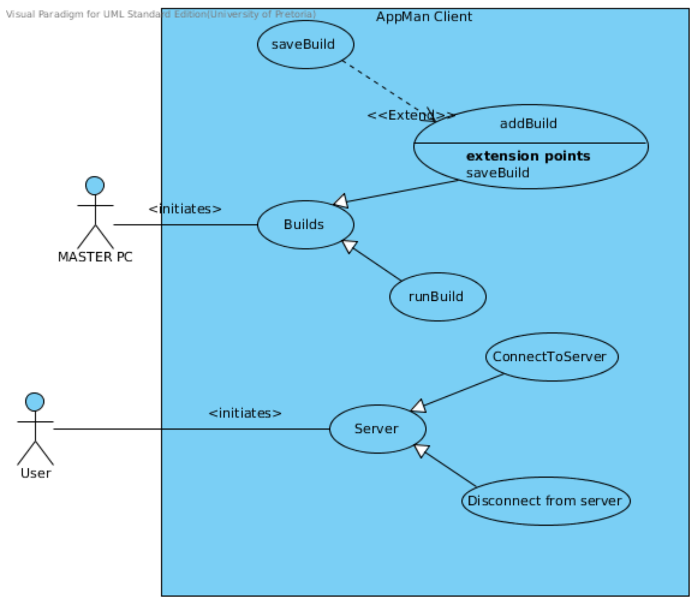
7 Functional Requirements

7.1 Use Case Diagrams

- AppMan



- AppManClient



7.2 Builds

- Builds must be addable to the master
- Builds must be updated on the slave computers once a change have been made
- Only changed files must be sent to the slave computers once a change has been made
- Easy to manage the builds that are on the master computer
- Builds must be able to have the following
  - Build Number
  - Directory
  - Name
  - Description

7.3 Copying

- Copying must be done over the network
- The copy of multiple files must be compressed to reduce the time it takes to copy a file

- Progress of copy must be shown on master computer while copying takes place

7.4 User Interface

- The user interface must be easy to use
- The user interface must make use of drag and drop capabilities to promote usability

7.5 Server & Client Applications

- The Applications must start on computer startup in a minimized form(e.g. in the tray)
- The Master machine server must start on application start
- Slave machines must connect to the master computer via the network once the application starts up
- Slave machines must prompt connection details once and then connect automatically afterwards
- The user interface must make use of drag and drop capabilities to promote usability

8 Glossary

- Build - An application build version that could potentially be distributed to slave computers
- Slave - A computer that will be controlled via a master computer
- Application builds will be sent to this computer
- Master - A computer that will control Slaves across a network
- Server - A machine waiting on the network for connections from other machines
- GUI - Graphical User Interface with which a user can control an application
- Project - This project. The distributed application manager

- Application Configuration - Environment variables that are specified when running an application