Developer’s Guide to Comnet V2

Mario Garcia, Michael Wallace

This is a rough sketch of what is to come for Comnet V2, the next innovation to its predecessor.

* Introduction
* Design Goals
* Basic Implementation
  + Basic Architecture
  + Overall layout
  + Useful features
* Basic how to Use
  + Digest Excel, xml, sheets // IN-DEVELOPMENT
  + Useful commands for logger // IN DEVELOPMENT
  + Setting up connections
    - Communication Protocols
      * UDP
      * TCP // IN-DEVELOPMENT
      * Serial
      * XBee // NEEDS-TESTING
        + WPAN
        + ZigBee
        + Digimesh
    - Set up port
    - Set up address
  + Sending packets
  + Receiving packets
    - Callback handling
      * Return results
  + Error Handling
* Intermediate Communications
  + Loose Coupling Procedures
  + Developing manual message handling
  + Extending Abstract Packet
  + Extending Abstract Queue
  + Linking Everything to one
  + Handling “Orphan” packages
* Advanced Techniques IN DEVELOPMENT
  + Improving overall performance
  + Advanced Loose Coupling features
    - Declaring custom allocators
      * Extending Allocator
    - Declaring custom data structures
      * Extending data structure interfaces
    - Tuning the Comms engine
    - Advanced Commands for Console log
    - Security Handling,
    - Custom Comparing
    - Custom Serialization
    - APIs for Communication handling
      * Specific sets for each protocol.
      * Defining your own protocols.
  + Future Works
    - Further Polishing
    - Compressing Packets
    - Testing Toolset
    - Publish-Subscribe Model
    - To Be Continued…
  + Conclusion
  + Index
  + Notes

Introduction

Before November of 2015, there was a need to provide a protocol layer for network communications using Wireless, UDP, TCP, and raw serial. By NATO STANAG 5602, there shall be a communications modules intended on IP based protocols, however datalink abstraction plays a crucial role in networks between various clients and users. For this, the creation of Protonet was important, in that it complied with most of NATO standardization of military protocols. Much was contributed in the work of Protonet Library, a Link abstractor and protocol parser were implanted to provide dynamic and rigorous transmission, along with the help of easily providing custom made packets without the need to redefine the entire packet.

Although the Protonet library has their merit, that is not to say that it went without problems, in fact there were multiple flaws to the design of the product that deemed it difficult, and clunky to use. For starters, the design of the library was not built to adapt to various features, in fact it was built too rigid to provide enough room for change. Of course, rigid design is indeed a definite trait; it does not work well when it comes to maintainability. For this, the library itself was not written with programmer-friendly features. The most notable features of this consequence comes deeply from how its API (Application Programming Interface) was designed, in that it required external scripts to provide updating of the library. The use of python scripts enabled developers to submit requests, so that they may be compiled into the library. The problem with this scheme is that it would take days, if not weeks, to recompile and redistribute the library with a new version. Many among these features can resemble to the early days of OpenGL, which gave all the work to the GPU kernel, instead of giving users and developers the freedom to define their own custom implementation, all while supporting and adapting the kernel to newer versions and updates. No more, however, a new library was sorely needed.

There were various needs to satisfy developers with easily integrating the library into their applications, all the while giving the library the ability to apply custom made packets. Comnet nodes which supply the endpoints between communications with others were in need of proper debugging and logging of performance. There was a deep need to provide a solid foundation in not only this, but also the adaptability of a product that can be maintainable, as well as rigid, for use in the long run.

Design Goals

CommProtocol (Comnet-V2) was specifically designed to meet these needs, along with giving the users and developers the freedom to both easily defined packets, dynamically, along with giving full modular control for extensive use. The library itself is capable of allowing users to simply declare packets without the need to submit requests to integrate them into the library. This is done through loose coupling, allowing users to inherit abstraction with their packets, and giving them the control on how it should be handled upon transmission and receive. Not only is this, but the library is itself fully customizable to allow creating new forms of nodes to handle any developer’s needs and desires.

The handling and design of the library was solely made possible through the use of object orientation and software engineering practices, coupled with research in data science. The communications pipeline undergoes rather advanced processes, yet they are handled through the use of simple tight coupling features that allow for greater performance through the use of internal structures. More on this later.

Basic How to Use