Software Server Model Using Socket Programming

Binoy Kurikaparambil Revi

Independent Researcher
binoyrevi@live.com

Abstract:
Software Server Model Architecture provides a robust solution to design the software system that serves complex requirements while maintaining the abstraction, security, maintainability, and scalability of a software application. With Software Server Model it is extremely easy to add, change or remove a server module without impacting on the software design of other server modules as each server module runs as an independent process. Monitoring, improving, and optimizing server modules is easy and induces less errors compared to monitoring, improving, and optimizing the complete software application as one single process.

Introduction:

Software Server Model using Socket Programming works on the idea that each major module runs as an independent process called server modules that execute a specific set of tasks to accomplish the overall purpose of the software and communicate with other server modules using sockets. If the server modules run on the same hardware unit, UNIX sockets are the preferred way to go. If the server modules run on different machines connected by network, TCP/IP sockets can be used.[3] However this comes with additional security overheads. Considering the fact that these server modules are running independently and have no idea about the status and state of other server modules, the Health Server Module plays a major part in the design.

Software Server Model Architecture

Figure 1 shows a simple sketch of Software Server Model Architecture. As we can see in the figure, Application Server Module 1, Application Server Module 2 and Health Server Module are 3 separate processes that run on the system.[4] The number of application server modules can vary depending on the complexity, number of software modules and size of the modules.

Application Server Module:

The Application Server Module contains 3 components.

1. Business Logic - This component is the requirement implementation of what the server module is expected to do. The implementation is encapsulated to the server module and communicates to system resources using Inter Process Communication (IPC) or system calls.[2]
2. Health Monitoring - Health Monitoring contains two tasks. The first basic task is to report the health status and state of the server module to the Health Server Module through the communication component. The second task is to get the health and status of the application server modules of interest from the system health status and state message packet issued by the Health Server Module. This is important as in most use cases, one application server module may be dependent on data, status, or state of another application server module. One such use case is start-up and default configuration.

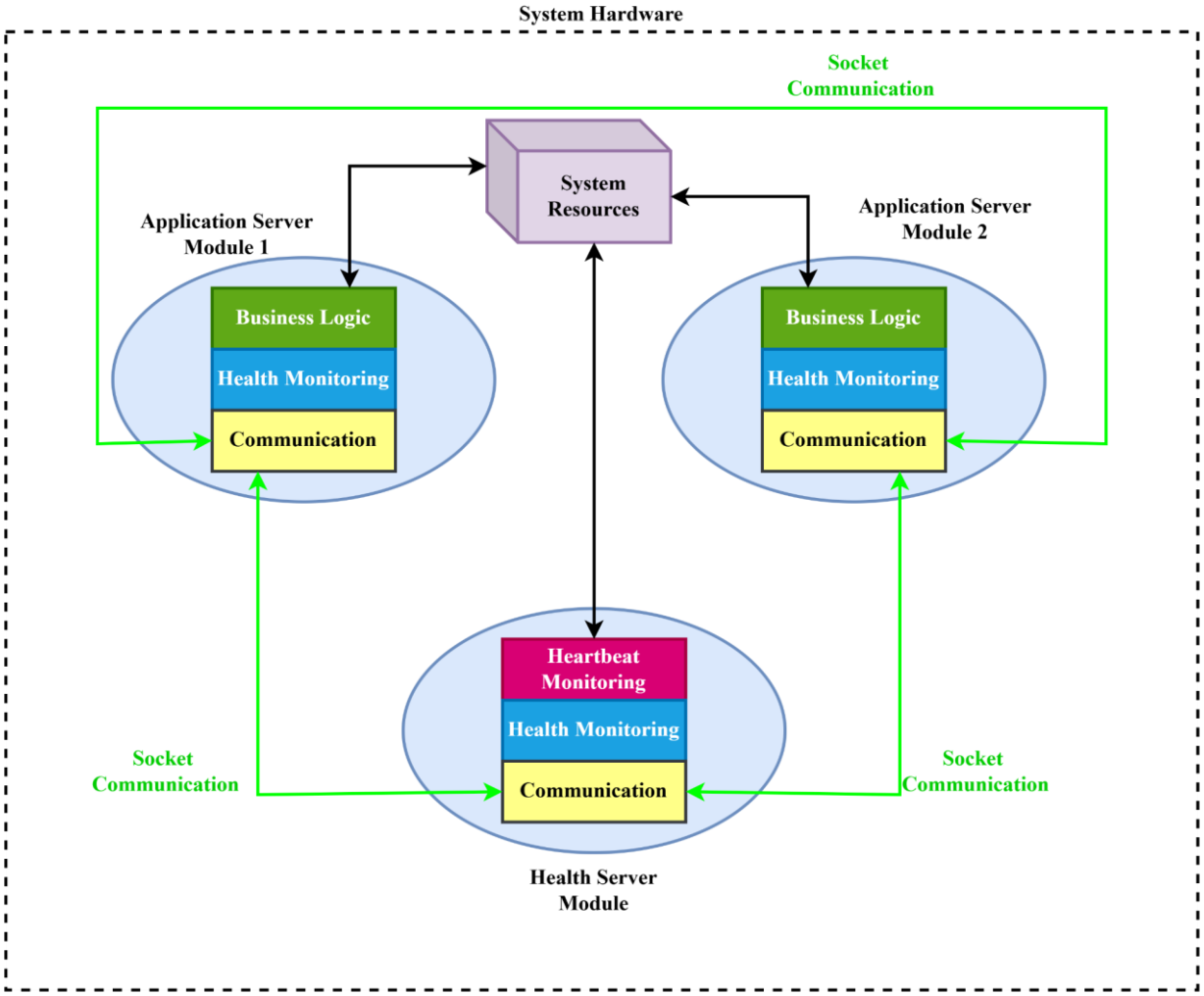

Figure 1: Software Server Model – Architecture

3. Communication Interface component - This can be visualized as a non-blocking thread that runs to send and receive messages using IPC socket between server modules.[1] Unix sockets are recommended to use for this purpose if all the server modules reside in the same hardware system and operating system.

The communication interface component of one server module will attempt to send a message to another server module only if the recipient has "Communication Ready" status as "True" on the latest system health status and state update.If the recipient has a 'Communication Ready' status set to "False", the message is either queued at the sender or error status is reported back to the business logic. The common status flags that may be used for communication are given below.

Server Module Running	True/False
Communication Ready	True/False
ACK Available	True/False
Protocol	String (This is defined by the application architecture)
Secured Messages	True/False

Application Flags	Int Array
Application Message	String

Based on the software requirements and design these flags shall be updated to meet functional and nonfunctional requirements.

Health Server Module:

The Health Server Module as the 3 components. The communication interface components and health components are the same as the Application Server Module. Instead of a Business Logic Component, the Health Server Module has Heartbeat Monitoring. This is critical to the Software Server Model Architecture. This module provides the following functionality.

1. The Heartbeat Monitoring component sends the heartbeat package to all the Application Server Modules in regular intervals. Heartbeat package contains the application health and status flags from all the application server modules binded to one package like a JSON package.
2. The Heartbeat Monitoring component communicates and reports overall system health and status. It reports if a server module is currently using system resources like writing to the file in the system disk or writing to a database or using a device.
3. The Heartbeat Monitoring component may also communicate system health and status to a remote device over the network using TCP/IP sockets by going through the communication interface component.
4. One important task of the Health Server Module is to format and package health and status information from all the Application Server Modules to a single message as per the end user needs and deliver it in user defined frequency. Example below demonstrates the overall system package in a JSON format.

```
{
  “Unix Time” : “1571580000000”
  “IP Address” : “12.234.232.12”
  “Application Name”: “Software Server Application”
  “Application ID”: “D4353535”
  “Token”: “43dfsw45453453hgwf4f345”
  “Applications” : [
      {
        “Server Module ID” : 01
        “Overall Status”: “Good”
        “Server Module Running” : True
        “Communication Ready” : True
        “ACK Available” : True
        “Protocol” : “App Protocol”
        “Secured Messages”: False
        “Application Flags”: “0A FE 2D 9B”
        “Application Message”: “Config Complete”
      },
      {
        “Server Module ID” : 02
        “Overall Status”: “False”
        “Server Module Running” : True
```

```
            "Communication Ready" : False
            "ACK Available" : True
            "Protocol" : "App Protocol"
            "Secured Messages": False
            "Application Flags": "0A 00 00 9B"
            "Application Message": "Comm Busy"
        },
        {
            "Server Module ID" : FF
            "Overall Status": "Good"
            "Server Module Running" : True
            "Communication Ready" : True
            "ACK Available" : True
            "Protocol" : "App Protocol"
            "Secured Messages": False
            "Application Flags": "00 00 00 00"
            "Application Message": "Health Server Module"
        }
    ]
}
```


Sharing System Resources

This is one of the most important tasks when designing a system using the Software Server Model. It is because, as Application Server Module runs as a process, it cannot predict the state and status of other Application Server Module processes. However, the Software Server Model provides an excellent mechanism to handle this in addition to standard Inter Process Communication(IPC) mechanisms.

Application flags are the key to manage the use of the system resources. Each bit or a complete byte in the application flag can be mapped to a resource that can be read as "In Use" or "Available". In some cases, it can have multiple states.When an Application Server Module finds that a resource is available to use, it will then communicate to the Health Server Module its "Intention to use" and if the Acknowledgement comes as "OK", the Application Server Module starts using it. If multiple requests are received by the Health Server Module at the same time or almost same time, it's the decision of the Health Server Module based on predefined priorities the resource is allocated.

Conclusion:

Software Server Model Architecture using Socket Programming provides excellent modularity and performance by running each major software module as a software process and maintaining communication between them using sockets. Benefits include:

- The Health Server Module provides an innovative solution for monitoring the health and status of all Application Server Modules and a dedicated mechanism to report the overall health and status to the user or system.
- The Health Server Module helps to resolve conflicts when multiple Application Server Modules try to use the same resource.
- From a development perspective it doesn't matter how each Application Server Module is developed. The programming language of one Application Server Module can be different from another. [1]
- The beauty of this architecture is that once the framework is developed, the framework along with the

Health Server Module can be reused in future projects which saves lots of time.

- Health Server Modules can be made capable of monitoring memory usage of Application Server Modules thus maintaining a real time profiling of application performance.

References:

1. Lin Yao, Ping Cao, Kezhu Song and Fuming Ruan, "A software design based on distributed architecture for seismic exploration system," 2010 Second IITA International Conference on Geoscience and Remote Sensing, Qingdao, 2010, pp. 567-570, doi: 10.1109/IITA-GRS.2010.5602687.
2. C. Wang and Y. Huang, "Design of the Socket-based IPC Messaging Platform," 2010 WASE International Conference on Information Engineering, Beidai, China, 2010, pp. 114-117, doi: 10.1109/ICIE.2010.318.
3. P. Xiao, Y. Li and W. Deng, "A Model of Distributed Interprocess Communication System," 2009 Second International Workshop on Knowledge Discovery and Data Mining, Moscow, Russia, 2009, pp. 276-279, doi: 10.1109/WKDD.2009.37.
4. N. Vun, K. Xu and Z. Foo, "A Modular Framework for Applications Development on Embedded Platforms," 2007 IEEE International Symposium on Consumer Electronics, Irving, TX, USA, 2007, pp. 1-6, doi: 10.1109/ISCE.2007.4382144.