Generic Application Programming Interface Design for Communication Protocols

Noman Sohaib Qureshi[1] and Dr. Malik Muhammad Saad Missen[2]

[1]Department of Computer Science and Engineering, University of Engineering and Technology, Lahore Pakistan (Gujranwala Campus-RCET)

[2]Department of Computer Science & Information Technology, The Islamia University of Bahawalpur, Pakistan

***Abstract*—The research proposal is to innovate a "Network Communication Application Programming Interface" which provides a platform to program network, communication applications on top of the API. The communication functionalities to be provided in this API are not limited to wired communication but it can be extended to program 'wireless' and 'wireless sensor networks'. For the API to be used in wireless sensor network programming it is necessary that the functionalities to be provided by the API are programmed in a programming environment and in such a way that it consumes fewer resources considering the limited processing capabilities and power of the wireless sensor network platform. The design of this API will be in such a way that the network communication functionalities will be programmed purely and completely in C language and these will be in the form of *Dynamic Link Libraries* as an extension to the operating system kernel. For providing the WSN programming mode in this API it is necessary to avoid any translation or simulation layer for the *'API core networking functionalities'*. It is necessary to deploy the API in form of DLLs to directly communicate with the operating system in its native language i.e. DLLs as windows itself is a collection of dynamic link libraries mostly programmed in Win32/64 SDK C/C++. This API can be used to program the communication layers of the windows based WSN applications and the upper layer graphical user interface and monitoring applications can be programmed in any of the high level language. Despite WSN network programming this API will be useful to develop applications related to '*Telephony*' or '*TV broadcast*' and the scope is extendable to vast area of network communications applications. The area of research for this MS(Computer Science) Thesis is to propose, design and develop such a Network Communication API. In this paper, at first stage, the design of the API is presented.**

***Keywords*—**Application Programming Interface, Dynamic Link Libraries, Operating System Kernel and Wireless Sensor Networks

This work was part of my postgraduate research in Thesis for MS (Computer Science) Degree with specialization in Computer Networks from Virtual University of Pakistan. The research was carried out under supervision of my Thesis Supervisor: Dr. Malik Muhammad Saad Missen.

Noman Sohaib Qureshi is with the Department of Computer Science & Engineering, University of Engineering & Technology, Lahore (Gujranwala Campus-RCET), (phone: +923009682419; e-mail: nomee_46@hotmail.com). Also Mr. Noman Sohaib Qureshi is a postgraduate research scholar in MS(Computer Science) at Virtual University Lahore Pakistan

Dr. Mailk Muhammad Saad Missen is with the Department of Computer Science and Information Technology, The Islamia University of Bahawalpur, Pakistan (e-mail: SaadMalik_2001@hotmail.com).

I. INTRODUCTION

By the research and survey in the area of *'Communication APIs'*, it is observed that most of the research for Network API is limited to a specific domain of communication functionalities e.g. it may be for wireless sensor network programming or may be programmed for wired or wireless communication specifically and usually which are developed for open source platforms. Research is necessary in network programming to develop a communications API whose domain is not limited to connectivity mode and which is useful both for wired and wireless communication and can be deployed for generic communication applications and that of sensor network applications as well. Another aspect found in current research is that designs and methods for internetworking between the connectionless and connection-oriented networks do exist [1]. However research for a generic API which provides these functionalities is necessary. Though general mode of wired or wireless communication APIs exist for commercial operating systems but the area of WSN API development is in its inception for commercial operating systems and an API for this purpose is necessary that may serve as a development core for the WSN and communication applications.

II. EXISTING COMMUNICATION APIS REVIEW

Today, there are several Network APIs some of which are proposed only, others are proposed and designed and yet some are proposed designed and developed. However one aspect is found common in the current APIs that they are domain specific and not generic in terms of application. In the section that follows a survey of the major existing APIs is presented as under.

"*A Network Application Programming Interface for Data Processing in Sensor Networks*" [2] is a proposed API whose domain is specific for sensor network communication applications. In the Rice University Technical Report TREE0705 the API is proposed and the initial design phase is presented however development and implementation is left for future research. It is a design of a higher level API which may provide middleware implementation in sensor network communication applications. This API proposes several node

based functionalities which are not supported by standard sensor network programming tools such as that found in the TinyOS [3]. The main focus of this proposed API is to overcome several deficiencies found in current WSN development tools which are tested on WSN simulators e.g. ns-2 etc. It provides abstract network services for communication in sensor networks. This proposed API supports three modes of communication in the sensor based networks i.e address based, region based and device hierarchy [4].

"*Smart Antenna API for SDR Network*" [5] is an API now under development phase [6]. This API is hardware specific and mainly 'software defined radio' centric and may not work on generic platforms and is not a suitable development interface for networks applications other than those involving SDRs communication. Also it communicates directly with the hardware without interaction with the kernel of an underlying system software e.g. TinyOS, Windows or UNIX. The team working on this API has proposed a hardware platform for employing the open architecture which remains unchanged for the implementation of the API [7]. The behavior of the hardware platform can be altered by the software API for specific functionality. The benefit is that it is the most important feature of this SDR API technology that a system update or an addition or deletion or modification of services can be performed with ease and without altering the existing hardware [8]. But the element common to other communication APIs is found in it as well that this Network API is also platform dependent and is not generic.

"COIP-K" [9] is a toolkit that can be used to implement proposed connection oriented internet protocols in UNIX environment. Although it is not completely an API but it provides the base in form of kernel functionalities for developing connection oriented applications which may interact with UNIX kernel. The *COIP-K* research focuses mainly on areas: data packets path in a connection oriented environment, resource requirement, resource reservation, termination of the connection and releasing the occupied resources. *COIP-K* research group has though proposed and provided base for the network applications to be build. However, it only works for connection oriented protocols and also is heavily dependent on *'Berkley software distribution 4.3'*. The scope of the *COIP-K* implementation is limited to BSD and UNIX kernel and the *COIP-K* research group have not presented the interface for generic application development at various heterogeneous platforms and also have no interface for wireless or wireless sensor networks.

III. GENERIC API DESIGN FOR COMMUNICATION

The main aim of this research is to propose, design and develop a generic *Network Communication API* that may be deployed to wired and wireless communication and which should be platform independent and may be applied to a wide variety of network projects/application's development ranging from simple communication software development up to wireless sensor networks.

By the extensive survey of existing literature and the designs of APIs for communication as presented in review section, the strengths and weaknesses of various APIs are counted in the proposal, design and development phase of this *'Network Communications API'*.

The basic architectural design of the API will that be of a bridge having many functionalities for both the source and the destination. Fig. 1elaborates the APIs position as a bridge.

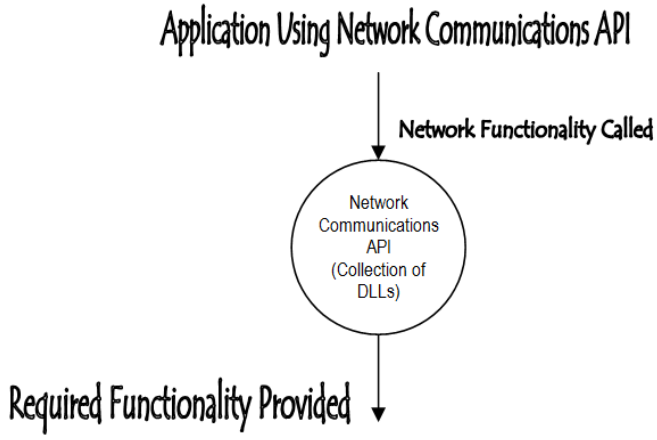

Fig. 1: API acts as a bridge with functionalities

The bridge shown in Fig. 1 may be extended to include few more essential entities as shown in Fig. 2.

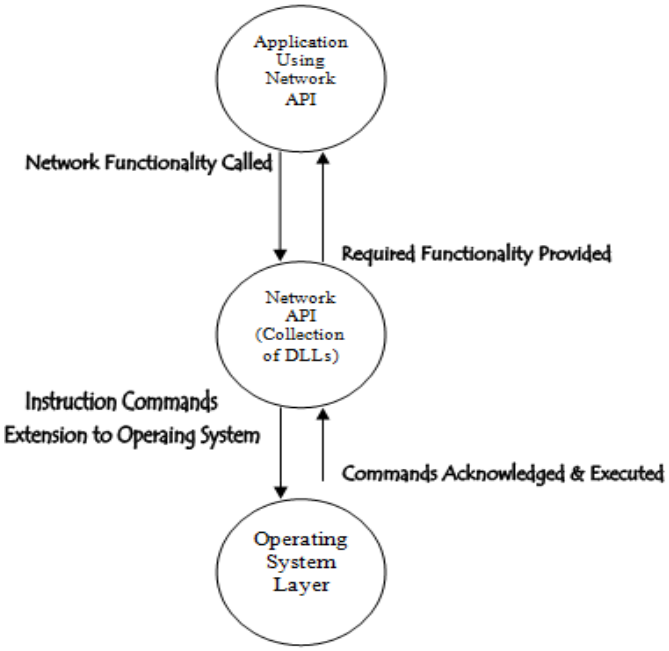

Fig. 2: API working as extended bridge

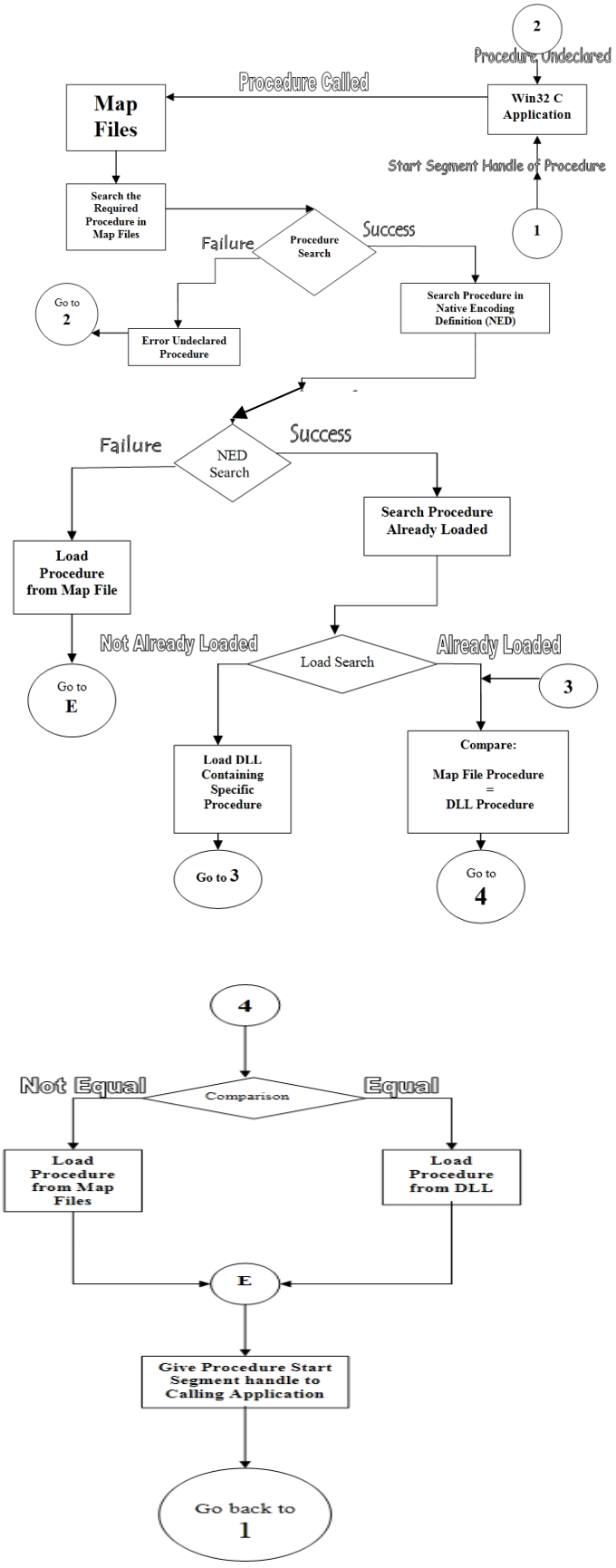

Fig. 3: Architecture Design for Generic Communication API

The overall architecture of the API is designed and presented in Fig. 3. The architecture contains the new aspect of map files. As its name suggests, map file are used for mapping called DLL functionalities to the API. It also has the capability of exception handling in already loaded procedures of DLL in memory by comparing it to the original backup of the procedure and if the loaded functionalities are altered from the original then the procedure is refreshed and called again from the backup using map files. With the development perspective the map files are the immediate header files in the API development architecture. These header files must not belong to development platform. Map files will be programmed and integrated in the API in such a way that it has higher priority than the dynamic link libraries in the messaging structure of system software. This priority is developed so that users own applications should not have any conflict with the heart of the operating system which is DLLs working in the kernel, graphic device interface (GDI) and user part of system software. Although, as it is necessary to override this default behavior so it is essential to handle the *Native Encoding Definition (NED)* architecture manually by the API which certainly means that API developers are now also responsible for *pragmas* in the data segment of the developed application. The map file priority based architecture is designed with *Native Encoding Definitions* and *Global Shared Data Segments*. In the architecture diagram win32 application refers to the C language application which is compatible with targeted 32-bit system software messaging structure and can have access to operating system kernel, GDI and user portion.

There will be three modes of communication in the design of the API that will be realized in the development phase which may be used to program various networked applications. The modes are presented as under:-

The first mode in design of the API is *'Blocking Connectivity Mode'*. The 'Blocking Connectivity' is just like a 'Telephone Call'; i.e., establish a connection on some number and the caller has to wait until the receiver responds to the caller as the behavior is shown in Fig. 3. This type of connectivity on networks can benefit to develop a *'Computer Telephone Exchange'* by using these 'Blocking Connectivity Mode' functionalities in the API. This connectivity should have the flexibility of connecting to specified numbers like in telephone exchanges. For WSN, it may be used for WSN platforms e.g. like that of a Wireless Robot which have one-to-one correspondence with the controlling server or for a remote WSN automated machine gun guarding a specified area etc. Also, this is not the end; by providing this blocking type connectivity on some number based system could result in many type of application to be built by using this API. In essence it is a connection oriented services mode in our API.

Fig. 4: Showing Blocking Connectivity Mode

The second mode of communication in the design of this research is *'Threading Connectivity Mode'*. This type of

connectivity breaks the network process into threads by which many block states for processes could be avoided as described in fig. 4; i.e. if a part of a network process is waiting for some event to happen to proceed further then the whole process should not be blocked. For this we use threads which are though efficient but require lots of code complexity and other challenges. This type of connectivity is very useful for developing applications for broadcasting e.g. *'Television Broadcast Networks'* etc. Threading connectivity mode in the *'Network Communication API'* will be ideal for developing communication software that involves broadcasting networks. In case of WSN if the server is attached to many WSN devices at a time and there are many wait states in processing then the threading functionality will be useful.

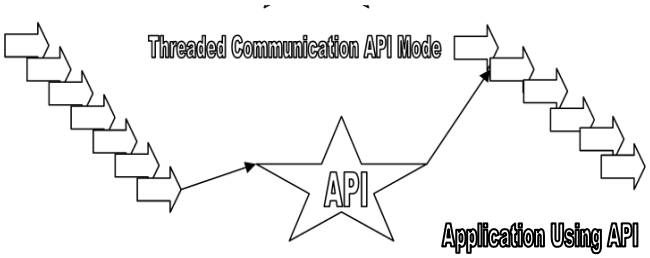

Fig. 5: Showing Process Breakup into Threads

The third mode of communication functionalities to be provided in our API is *'Asynchronous Connectivity Mode'*. This behavior is the one in which the request to receive a connection can be cancelled. Also after initiating a connection request it cannot be in block state like 'Telephone in Blocking Connectivity'; rather in this mode the application is free to perform any other activity. It is a connectionless service in our API.

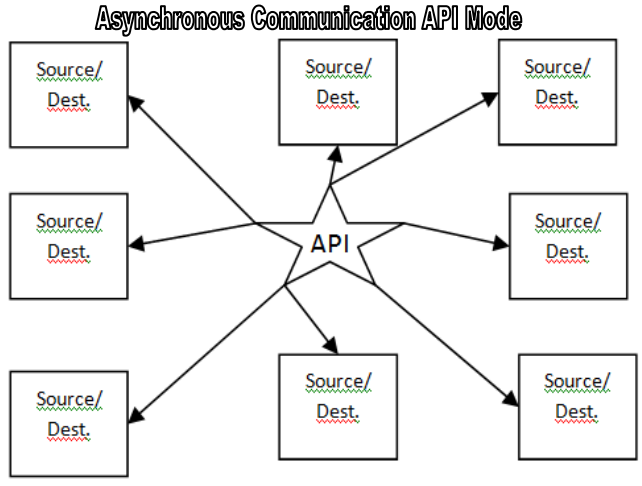

Fig. 6: Showing Asynchronous API Mode

This type of connectivity is useful in *"Chatting"* or *"Conferences"* because we can cancel the request any time and on the same time we can do other activities as well. In WSN platform connectionless functionality is useful when there are more than one server controlling the WSN device and the device is not required to be blocked or bound in one-to-one connection oriented protocol like described in first goal of this API. Usually home appliances WSN devices and the WSN platforms which do not require enhanced secure mode of communication may be programmed by this functionality.

IV. CONCLUSION

The API is proposed and designed is this paper. The design work is done in such a way to accomplish as much as possible the goal to keep the API *'generic'* and not application or hardware specific. It may be deployed to a wide variety of areas of application. This paper presents the three major modules in the design phase relating to this API. However the API is divided into many modules as sub modules of the three major. Architecture and state transition diagrams are designed that will be programmed in development phase. The care has been done at each stage that anything that violates the precondition of platform independency and interpretability to other platforms must be intact. The design also provides exception handling services by means of architecture module design. Several checks are made to ensure that all modules of the API function correctly and if any unprecedented event occurs the backup is restored from the static mapping that will be maintained in form of map files.

REFERENCES

[1] M. Veeraraghavan and M. Karol, "Internetworking Connectionless and Connection-Oriented Networks," *IEEE Communications Magazine*, Dec. 1999, published as a Selected Paper from *IEEE BSS'99*.

[2] R. Wagner, J. R. Stinnett, M. Duarte, R. Baraniuk, D. B. Johnson, and T. S. E. Ng, "A Network Application Programming Interface for Data Processing in Sensor Networks," Rice University, Tech. Rep. TREE0705, Jan. 2007.

[3] TinyOS Community Forum. http://www.tinyos.net

[4] R. Wagner, M. Duarte, J. R. Stinnett, T. S. E. Ng, D. B. Johnson, and R. Baraniuk, "A network API-driven survey of communication requirements of distributed data processing algorithms for sensor networks," Rice University, Tech. Rep., 2006.

[5] Namkyu Ryu, Taeyoul Oh, Seungwon Choi and Soonjoon Park, "Smart Antenna API for SDR Network, " Proceeding of the SDR 05 Technical Conference and Product Exposition. 2005

[6] Seungheon Hyun, June Kim, Seungwon Choi, Lee Pucker, and Bruce Fette, "Standardizing smart antenna API for SDR networks", Software Radio. Vol. 18- P. 34-40, 2007

[7] J. H. Reed (*Virginia Tech*), "Software Radio: A modern approach to radio engineering", Prentice Communications Engineering and Emerging technology series 2002.

[8] J. Mitola, "*The software radio architecture*," IEEE Communication Mag., vol. 33, no. 5, pp. 26-38, 1995

[9] C. Cranor and G. Parulkar. An Implementation Model for Connection Oriented Internet Protocols. *Journal of Internetworking*, 4(3):133–157, 1993.