2. LITERATURE REVIEW

2.1 INTRODUCTION

In this modern arena everything relies on Computers. Users expect speed and efficiency among computers. This both features depend upon the type of machine and the type of operating system installed in that machine. The system type is based on the cost and the standards of Manufacturers who build the machine. But the performance of Operating system mostly depends upon the features provided by the loaders, as they are considered as the init of the operating systems. Every loader has its own specifications. Even such loaders possess their own abilities they have to rely BIOS as their root because is capable of loading every loaders all of operating systems. Thus their features are not a part when BIOS takes such a role. No loaders are BIOS independent. Moreover the operating systems in-cooperating such loader takes a huge disk space when installed. And if to be a portable one the disk space consumed will be more. So the design concentrated on developing a dynamic loader which removes the dependency of BIOS and also a portable Linux operating system that is loaded in a USB stick. The focus of this chapter is to provide an important summarization of primitive to recent advancements in the area of Opensource Operating Systems as well as loaders.

2.2 STATIC BOOT LOADER TECHNIQUES

Operating system could not run without support of boot loader, so a lot of boot loaders for different architectures has been designed and used. The beginning of the boot-up process varies depending on the hardware platform being used. However once the kernel is found and loaded by the boot loader, the default boot-up process is identical across all architectures. When the system is powered on, it launches the first stage machine code on the
Master Boot Record (MBR) of the flash memory, named a boot loader. The boot loader loads itself into memory and then yields control of the boot-up process to it.

The boot techniques originated with a design of a bootstrap algorithm for MIMO systems. **M.S. Ahmed and N. Sait in September 1989,** proposed this bootstrap algorithm for the self tuning control of MIMO Systems. They split the non-linear problem into 2 linear ones and the state estimation is carried out by assuming that the system parameters are known. This algorithm is used for online identification and control [12]. This great success of boot design motivated the researchers to go on developments with the boot design structure.

Due to these developments, **Stephen A. Rogers and Leonard E. Schulwit in the same year** developed a bootstrap for a Distributed System (MULTIBUS II system) as a set of connected hosts. In this technique bootstrap parameters are located at a number of hosts and load the needed parameters when needed with a request or a call [13]. Better flexibility and a good start was achieved with the help of these boot loaders. This attracted the researchers on electrical field to implement the boot concept in the field of Electronics.

As a result **in 1992 S. Xiao, R.Y.V.Chik and C.A.T. Salama introduced** self-booting technique to increase the output resistance of MESFET [14]. This was a considerable achievement in the field of electronics. The boot sectors that are applied in the Distributed Systems are vulnerable to viruses called Boot sector viruses, which makes the Distributed systems in a boot failure. So that entire systems get collapsed.

By taking this under consideration **Gerald J. Tesauro, Jeffrey O. Kephart, and Gregory B.Sorkin, IBM T.J. Watson Re-search Center used** a Neural Networks scheme on boot sectors to prevent the attack of boot straps from boot-sector viruses [15]. Not only in the world of computers but in the field of embedded systems, boot loaders made a remarkable part.
FranCois Theberge and Ravi R. Mazumdar in 1996 proposed a bootstrap procedure to influence the rate in the loss model of a large capacity ATM switch which can support both constant bit rate (CBR) and variable bit rate (VBR) traffic so that the number of connections were in more admit [16]. While considering the features of boot loader every departments of science implemented its concept at various fields.

In the same year Yair Shimshoni and Nathan Intrator introduced Bagging technique using bootstrap was implemented for aggregate estimators in Seismic survey. Bootstrap samples were the major technique implemented here [17]. Again the boot loaders occupied the field of electronics via the area of amplifiers.

C. Hoyle and A. Peyton in 1999 used shunt bootstrapping technique to improve bandwidth of trans impedance amplifiers [18]. In twentieth century every computer based applications required some sort of boot algorithms or programs to init their functions.

In 2002 E.G.M. de Lacerda and A.C.P.L.F. de Carvalho proposed a Bootstrap technique that was implemented in Genetic Algorithms to detect true errors using bootstrap data sets [19]. All such boot techniques followed a same procedure and concept of initiating actions either in software or hardware.

But Ronald G. Minnich from Los Alumos National Lab in 2004 used their own bootstrap. This innovation reduces the risk of locating, verifying and loading a new OS image because here OS boots and the other OS systems work. This tricky boot is implemented by opening and validating a new image and moving it to the kernel space, now prepare for hardware reboot and load the new kernel to the existing running kernel, and entirely move to the new kernel [20]. This rebuilds the concept of tailoring the boot strap according to the requirements of the application. Even boot mechanisms have all such advantages users still have to rely on the boot times. Some systems took boot time a lot to load the required functions.
Tony Benavides, Justin Treon, Jared Hulbert, and Willie Chang in 2007 implemented a Hybrid-Execute-In-Place Architecture that reduced the Boot Time to some extent [21]. Such a design provided more efficiency in boot loaders. The arise of VLSI technology made the development of embedded loaders into a single chip.

According to Nan Zhang, Jijun Ma, Jian Chen and Tianzhou Chen (2008) loading of boot loader took from a Scratch-Pad Memory (SPM), which is small, isolated and located on a single chip. This made the booting speed to some extent. They implemented this technique in a Novel loading process. In this technique boot loader initializes the hardware and the complicated functions are performed by the SPM [22]. But this chip based loaders faced serious disadvantages. If we need to modify this loader it needs a flash or a new chip. So researchers focused on a programmable loader.

Due to this research in 2008 Xiaofeng Wan, Hailin Hu and Xiaojun Zhou designed a boot loader for an embedded system, the hinge connects hardware and OS, is made up of OEM startup code and main code mostly. At the same time, some configuration files should be edited, such as sources, makefile and Boot.bib. The designed boot loader can also run in the video surveillance system successfully [23]. The programmable loader usually consists of files which can be modified and designed later. Developments in loader made a single loader to initiate the main as well its peripherals also.

In 2008 Pei Ke, Zhang Gang and Li Fu-jiang designed a Boot loader with multiple communication ports to load an operating system in an embedded system with complex embedded microprocessor inner resources and plentiful peripheral devices. The designed boot loader also loads the linux operating system kernel to the ARM & DSP dual-core embedded microprocessor system development platform to download the file via three common communication interfaces (UART, Ethernet0, USB port). This lead to the
development of boot loader on the basis of hardware platforms [24]. In recent years network
based loaders came into existence.

Guanfei GUO, Yaoxue ZHANG, Yuezhi ZHOU, Laurence T. YANG, Li WEI, Pengwei TIAN in 2008 designed a booting process in a transparent environment. Here the OS for the client is loaded from the server from one of the virtual disks, the client being loaded without any local memory [25]. Every research focused on loader performance but not on its security. Even at loading stages hackers can enter into the system.

In order to overcome this Junkai Gu and Weiyong Ji in 2009 proposed a secure bootstrap based on trusted computing technology. Here each time when the kernel is booted, the integrity of files and code are measured first. If this test passes the control goes to Master Boot Record (MBR) and performs the ordinary bootstrap process. Otherwise the illegally modified all files will be recovered. For this secure bootstrap, the integrity of files that load the OS and the dynamic integrity of OS Loaders were considered. They ensured the security of system by providing a smart card to the users of system. If the card validates to be expired or wrong the system goes to halt state [26].

In 2009 portable loaders were incorporated in web GIS applications. Zhou Di-bo and Pan De-lu proposed a techniques to use web GIS application, it requires complicated configuration. For saving the energy and time needed by the installation, configuration and maintenance of web GIS, the pre-configured plug and play web GIS is developed. This plug and play web GIS present in the USB flash drive operating system is booted through BIOS and can use the web GIS application [27].

A modified form of secure boot was proposed by Hans Lohr, Ahmad-Reza Sadeghi and Marcel Winandy in 2010 as a Patterns for Secure Boot and Secure Storage in Trusted Systems. In this secure boot they verified the integrity of the software before boot starts and access the resources. They used a Root of Trust for Measurement to execute before booting
which loads and measures the program code of the subsequent Bootstrap Module, and verifies the integrity of the code. If it violates the Trust, it stops the execution and the system is halted [28]. Very often boot extended its hands in the area of Genetic networks.

**Shuhei Kimura, Koki Matsumura and Mariko Okada-Hatakeyama in 2010** used bootstrap methods with interference method using LPM to reduce the number of false-positive regulations in Genetic Networks [29]. The bootstrap techniques protected the operated system from vulnerable attacks.

**In 2010 Jie Zhang and Fengjing Shao** implemented a boot loader module based on the new high-security operating system with internal networking structure netOS-I. This module is capable of initializing the hardware environment of new computer architecture sCPU-dBUS, establishing the memory mapping, setting up the environment for netOS-I, and then loading and starting up netOS-I [30]. A modified form of operating system is developed using a portable loader in 2010.

**Guodong Li and Hu Chen** developed a new high level security portable system based on USB key with fingerprint is designed to protect the host computer, where the fingerprint-based authentication security is portable custom-build software environment, present in the USB device. The authentication process will be done inside the USB device while booting the USB operating system itself and the booting process is through BIOS [31]. This loaders can also be incorporated in any portable storage devices.

**In 2010 Anil Kumar Kama** proposed that an External USB hard disk can be used as an installation resource for various operating systems and also as a bootable media. It is the best substitute for laptop to carry various operating systems along with other personal data. The external USB hard disk is loaded through BIOS for both installing and booting an operating system [32].
Geng Qingtian, Sun Zhanchen, Zhao Hongwei and Gu Jianhao in 2011 made modifications in U-boot (Universal Boot Loader) to support for S3C2440 processor. They transferred the support files of U-boot S3C2410 so that it can support the hardware platform based on S3C2440 processor. In this U-boot transplantation U-boot 2011.03 is used [33]. In 2012 modified loaders with encryption, data integrity is developed to ensure high security to operating systems.

Kuan-Jen Lin and Chin-Yi Wang in 2012 extended TPM BIOS interrupt calls to support RAS encryption and decryption and accessing the NVRAM. They enhanced the security of the encrypted BIOS password using TPM RSA engine instead of conventional encryption and added techniques for storing the encrypted password to TPM NVRAM so that an attacker cannot clear it by removing the battery. They used TPM SHA-1 engine to verify data integrity of the full MBR and determine if the booting continues [34].

2.3 OPERATING SYSTEM TECHNIQUES

Several techniques were introduced in operating systems form early itself to maintains the speed and consistency of operating systems. Every operating system even fully functional primarily relied on the speed and performance of loaders. The following specifies the various researches that took place in the development of operating systems. The first remarkable achievement in the field of operating systems is in FTM.

In 1996 Gilles Muller, Michel Baniitre, Nadine Peyrouze and Bruno Rochat developed a fault tolerant system FTM (Fault-Tolerant Multiprocessor). It offered fault tolerance to user applications. In FTM we need only compiling the source code without having to modify it. The main objective of the FTM design was to implement a low cost fault-tolerant system that could be used on standard workstations. This FTM is based on high-performance stable storage implementation, called Stable Transactional Memory (STM)
to increase its performance. It preserves the benefits of micro-kernel based OS. It does not introduce RPC performance degradation. Reducing server checkpoint size assumes a cycling behaviour and inner calls are lost in the event of a crash [35]. Every operating system needs to be built in a different manner in accordance with various platforms. To overcome this Strata technology was developed.

Kevin Scott and Jack Davidson in 2002 introduced portable Safe Virtual Execution (SVE) framework, called Strata. Strata are designed to be ported easily to new platforms and to date has been targeted to SPARC/Solaris, x86/Linux, and MIPS/IRIX. This portability ensures that SVE applications implemented in Strata are available to a wide variety of host systems. Strata also afford the opportunity for code reuse among different SVE applications by establishing a common implementation framework. It does not rely on special operating system services, compilers, or libraries. The user does not have to learn a new domain specific language in order to write security policies. When source code is available, a variety of static analysis techniques can be used to locate buffer overflow vulnerabilities, API misuses, and other programming errors that may result in security vulnerabilities [36]. Even operating systems come from different vendors, researchers focussed only on Linux operating systems because it is opensource and freeware.

Sergio Perez, Joan Vila in 2003 developed a RT Linux (Real-Time Linux GPL) which is a small, deterministic, real-time kernel that handles time-critical tasks and runs Linux as its lowest priority execution thread. However the real-time tasks in RT- Linux cannot make use of Linux services and, in particular, TCP/IP networking. But this mode of system is highly reliable because it is designed to take advantage of sharing memory. The Linux kernel and Linux user processes are all considered as the “Linux task” by RTLlinux and for RTLlinux the “Linux task” is the lowest priority task. This embedded systems real-time tasks cannot make use of linear services and, in particular, TCP/IP networking [37].
Evolution of more number of operating systems created a demand of security in these operating systems. This lead to the development of Trusted Systems.

In 2003 Yung-Feng Lu, Chin-Fu Kuo and Chung-Ihsin Chen introduced trusted operating system concept in OpenSource Operating Systems. This trusted operating system provides the basic security mechanisms and services that can let a computer system to distinguish, and separate and protect security data. Trusted operating systems split their own resources (such as file, print or network access) into compartments, or “sandboxes,” and allow only certain end users, administrators or applications into those areas. It implemented security requirements for commercial-off-the-shelf (COTS) general-purpose multilevel operating systems in networked environments containing sensitive information [38]. While taking every operating system it took more primary and secondary memories for proper functioning. So there was a demand for light weight operating system.

In 2004 Adam Dunkels, Bjorn Gronvall and Thiemo Voigt developed Contiki, a lightweight operating system with support for dynamic loading and replacement of individual programs and services. Contiki is built around an event-driven kernel but provides optional preemptive multithreading that can be applied to individual processes. Contiki provides a dynamic structure which allows programs and drivers to be replaced during run-time and without relinking. The Contiki kernel consists of a lightweight event scheduler that dispatches events to running processes and periodically calls processes polling handlers. All events scheduling in Contiki is done at a single level and events cannot preempt each other. The Contiki kernel only provides the most basic CPU multiplexing and event handling features. The rest of the system is implemented as system libraries that are optionally linked with programs. The porting process of contiki consists of writing the boot up code, device drivers, the architecture specific parts of the program loader, and the stack switching code of the multithreading library. The kernel and the service layer does not require any changes. The
Contiki kernel consists of a lightweight event scheduler. The Contiki kernel uses a single shared stack for all process execution. Increased energy is spent in interpreting the code—for long running programs the energy saved during the transport of the binary code is instead spent in the overhead of executing the code is a serious threat in Contiki [39]. This was the first light weight operating system that came with full features.

Jane Jayaputera, Iman Poernomo, Heinz Schmidt in 2004 proposed a methodology for run-time verification of reliability and availability properties for distributed architectures. They implemented a .NET-based system for monitoring contracts, which was built upon the Windows Management Instrumentation (WMI) framework. They adapted Probabilistic Computational Tree Logic (PCTL) to design-by-contract, which becomes easier to systematically test system deployments against timing and probability constraints and to improve implementation performance [40]. For a fully functional operating systems played a major role.

Yong-Seok Kim, Hee-Sun Kim and Chang-Goo Lee in 2004 developed a home control network system consisting of components such as embedded home server, the USB home controller, and the CDMA module. The USB home controller controls various domestic appliances, and the home server manages the USB Home Controller and also supports the SMS and wired/wireless internet using a network device and a CDMA module. The embedded home Server can manage each device of the device driver because it has embedded OS porting. It is also developed to manage home networking through the web browser by an outside client on the web server and the CGI. This provided a safe and smart security to home via an USB controller with an inbuilt embedded operating system [41]. Earlier researches developed a way for light weight operating system for computers. But 2005 made a revolutionary change in the arena of embedded systems.
In 2005 Kwang-il Hwang, Jin-woo Kim, Jeongsik In, and Doo-seop Eom proposed an event-driven operating system called the **Embedded Lightweight Operating System (ELOS)** including fault tolerant self-configuration combined with a multi hop ad hoc routing protocol suite. The design consists of a conditional preemptive FCFS scheduling method with a guaranteed time slot and fault tolerance is reinforced by supplementing semi-auto configuration using wireless agent nodes. The ELOS uses a packet buffer for efficient packet handling. ELOS has the ability of self-organization on support of the MAC and routing protocols. Ubiquitous network nodes are larger in size in such systems and are less autonomous [42].

Even portable operating systems exists they can be easily deleted and not configurable to various platforms. But **Nishkam Ravi, Chandra Narayanaswami and Mandayam Raghunath in 2007** identified and classified the set of attacks that a malicious host machine can launch on the pocket hard drive particularly USB. They described a set of software security mechanisms that the pocket hard drive can use to verify the host machine’s integrity. Here the pocket hard drive carries a base OS that loads during bootup. An encrypted partition contains user data and applications, which the base OS decrypts to restore a user’s session on the borrowed computer. The borrowed platform’s integrity is established before the base OS decrypts the sensitive partition. This model focused on attacks that are created by technical experts intended to attack pocket hard drive via internet [43]. Every shutdown mechanism of operating system wants to load all essential drivers on start up. It is more time consuming and provide discomfort to the users.

**Ying-Wen Bai and Huang-Te Hsu in 2007** proposed “Suspended to RAM” technique to replace the “System off” method to perform an instantaneous turning-on system just like some other consumer electronics. They expressed that the system memory is a suitable storage medium to store the OS and by using the high speed of the system memory...
the OS is reloaded in a short time and thus an instant system turn-on is achieved. In this model a fully utilized system boots into the OS in 3 seconds, with a power consumption of less than 0.7 watt [44]. This led to the development of hibernation technology in operating systems.

The boot loader designed by Xiaofeng Wan, Hailin Hu and Xiaojun Zhou (2008) connects a S3C2410X CPU and Windows CE 5.0 RTOS for the purpose of running Electronic Power Steering systems. Here the boot loader is made up of OEM code and main code mostly and initializes the hardware and download the OS image to the memory. NAND flash boot loader also called as Nboot is adopted in this design [45].

Dolev S and Yagel R in 2008 proposed a concept of Self-Stabilizing Operating Systems. They developed this based on two approaches:

- The first approach is based on periodical automatic reinstalling of the operating system and restart.
- The second reinstall the executable portion of the operating system and uses predicates on the operating system state (content of variables) to ensure that the operating system does not diverge from its specifications.

By using these 2 concepts we could obtain self-stabilizing operating systems. The performance of such an operating system can be tuned according to the time period required and chosen for recovery. Frequency of the checks may be balanced against the desired recovery speed. The tests included the same task being run by increased number of processes [46]. Servers in Linux need to be handled careful which are fully instructional. But 2008 gave birth to a AWK linux, an instructional desktop operating system.

In the same year Yung-Pin Cheng and Janet Mei-Chuen Lin (2008) proposed well-known instructional operating systems are complex, particularly if their companion software is taken into account. It takes considerable time and effort to craft these systems, and their
complexity may introduce maintenance and evolution problems. In this paper, a courseware called Awk-Linux is proposed. Awk-Linux can be crafted relatively more easily and it does not depend on any hardware simulator or platform. The basic hardware functions provided by Awk-Linux include timer interrupt and page-fault interrupt, which are simulated through program instrumentation over user programs. Course projects based on Awk-Linux provide source code extracted and simplified from a Linux kernel. Results of this study indicate that the projects helped students better to understand inner workings of operating systems [47].

To obtain the information of intruders in network a command line tool called honeyd is used. But Chao-Hsi Yeh and Chung-Huang Yang in 2008 constructed a GUI for honeyd (is an open-source honeypot) which is installed in a USB stick. A honeypot is a type of information system that is used to obtain information on intruders in a network. This was used in front of a firewall so that it can serve as an early warning system. As Honeypot is a command line interface and difficult for the beginners they developed it as a GUI. They deployed the designed system in a campus network and presented an analytic result of a 60-day period with a Web-based data analysis system [48]. More number of advancements was incorporated in the field of operating systems on the followed years.

Dongkyun Ahn and Gyungho Lee in 2009 proposed a simple finite state machine to track usage of stack frame locations at a fine granularity of 2-bytes. Such a fine grain protection is necessary to distinguish adjacent stack frame locations, which allows detecting abnormal memory operations even in real mode running of a boot loader. This StackLock mechanism is used to detect illegitimate writes in run-time stack area. This model was proposed because Conventional stack layout, in which local variables for user input and control of data such as return address are saved close to each other, is often the root of the attack vulnerability to malicious programs [49]. In these years both shareware and freeware provided more number of features to the worldwide users.
Daoxun Xia, Xiaoyao Xie, Yang Xu in 2009 proposed Web GIS using opensource software and opensource operating system. It is constructed on Open Operating System Linux and open-source application software such as Apache, Geo Server, PostGIS/MySQL. The system developed with open source software is very suitable for organizations that have limited financial budget for spatial data sharing, especially in developing countries [50]. Every operating system runs on kernel. This kernel should be made secure.

In 2010 Stefan Nurnberger, Thomas Feller and Sorin A. Huss proposed a secure micro kernel called Ray keeping security in mind. It was developed from scratch with new features of security such as memory gifts and behavior deviation detection. This approach avoids the interruption of non-reentrant code. The Ray micro kernel features IPC based on signals that are started as separate threads. As a fire-and-forget type message, no implicit expectations exist about the handling of messages and their protection using mutual exclusion. Urgent events can apparently not be signalled immediately when an old event is still being processed in RAY [51].

J.C. Chiu and T.L. Yeh in 2010 proposed a new concept called IRES. The framework supports reconfigurable computing architectures, based on traditional central processing unit and the reconfigurable field programmable gate array, and composed of the integration linker, the boot loader, small task-oriented operating objects and the hardware management unit. The integration linker enables the IRES to link hardware net-list files and tasks into one execution file, called the executor, constructed with the boot loader, the task-oriented operating kernel, the application tasks and the accelerating hardware functions. When the executor operates on the target-embedded environment, the implicit hardwire-call will be supported to invoke hardware functions in the task codes [52].

In general primary memory loads only the essential components of an operating system. But in 2010 Julian James Stephen, designed a small operating system in which the boot
loader program loaded the kernel of the operating system into the main memory for execution. [53]. Operating systems also played a major role in data acquisition system.

**Fabrice Leroux, T. Saint-Laurent in 2010** proposed an alternative approach to Tore Supra data acquisition system. It is based on a based on an open source Linux operating system with a diskless system. The performance obtained is fair equal to the already built Tore Supra data acquisition model. The use of open source solutions in this approach reduced the cost of data acquisition systems [54].

**Ji-Hyoung Ryu, Hee-Sun Kim and Chnag-Goo Lee in 2010** developed USB based terminals in U-Learning technique. The USB terminals, used in this system, are inexpensive, and it is easy to maintain their performances. Also, this system solves the problems of security in a local network and provides guaranteed QoS (quality of service). They used the OSGi specification as a middleware. It supports the discovery mechanism of the USB terminals, maintenance and administration of the system [55]. Even operating systems exists users aimed on portable operating systems, so that they can meet their needs wherever they go.

**Chung-Huang Yang and Pei-Hua Yen in 2010** created a live USB so that target hosts can boot from the USB which contains a functional operating system with tools for forensic discovery. This helped the investigator to collect digital evidence of target computer after any sort of hacking incident is occurred. Moreover they concentrated on open source applications to build this system which reduced the cost to a great extent [56].

**Dong-oh Kang, Jinho Yoo and Jeunwoo Lee in 2010** propose a new remote USB architecture by adopting the virtual connection manager which sustains the virtual connection among the virtual machine and remote USB devices. With the proposed remote USB architecture, users of the virtual machine use remote USB devices seamlessly without
noticing live migration of virtual machines [57]. These portable Live USB systems can be incorporated in systems of any architecture.

**Gaohua Liao, Quanguo Lu and Weizhong Zhang in 2010** extended USB host controller of embedded computer by system bus interface based on 32-bit microprocessor Strong ARM* SA-1110 to avoid hardware design difficulties caused by using the traditional methods of DMA. This produced a simple hardware circuit, the design of low-risk, easy debugging, as well as drive under Windows CE easily achieved [58]. If operating systems were not built to their platform users can integrate the operating systems using virtualisation.

**Pablo Pessolani and Oscar Jara in 2011** proposed a user-space multiserver operating system which is a modified form of MINIX Opensource operating system, running on top of a middleware-based virtual machine with simulated hardware constructed from services provided by a host operating system usually a Linux. They made this work is to develop an approach of a Multiserver OS running in user-space using OS-based virtualization for distributed processing and SSIDOSs [59]. Recent techniques also integrated operated systems with forensics.

**Dija S, Balan C, Anoop V and Ramani B in 2011 proposed a concept in** the recovery of Bitlocked Drives both in Live and Offline Forensics. They introduced different ways to recover fixed or removable storage media drives, bitlocked in USB-only mode [60].

In **2011 Teoh Tiong Hong, Sarwar Zeeshan and Mohd Zaid Abdullah** proposed architecture for USB 2.0 which is more power efficient using clock gating techniques as well as a fine grain power gating approach based on data traffic. The open source USB 2.0 device controller soft-core which is maintained by the OpenCores Organization will be used as a base design in this model. About 88% power reduction is achieved by the new modified design with minimal area and performance penalties [61]. Integrating network security
applications in operating systems made the operating systems to protect the networks through a server.

**Deng Yiquan in 2011** introduced some new methods of protecting network security in Linux Operating Systems which set the right to accessing the customer machine and use firewall technology. This methodology reduced the use of external applications in Linux Operating systems to ensure reliable protection in Networks [62]. Operating systems also made a role in IDACS.

**Manivannan M and Kumaresan N in 2011** developed a system using ARM9 Processor portability with Real Time Linux operating system (RTLinux RTOS) for developing on-line Interactive Data Acquisition and Control System (IDACS). This made the system more real-time and handling various processes based on multi-tasking and reliable scheduling mechanisms. Web server application is ported into an ARM processor using embedded C Language [63]. Still a number of operating systems exist while making a small comparison in operating systems it provided the following results.

**Yan Cui, Yingxin Wang, Yu Chen and Yuanchun Shi in 2011** used 3 operating systems, Linux, Solaris and FreeBSD to systematically evaluate and compare their scalability by using a set of highly-focused microbenchmarks for broad and detailed understanding their scalability on an AMD 32-core system. They used system profiling tools and analyze kernel source codes to find out the root cause of each observed scalability bottleneck. Their result concluded that the scalability of an application on any of the investigated operating systems running on a system of a larger number of cores [64].

**M. G. D’Elia and V. Paciello in 2011** evaluated a test on the performance issues on Linux, Ubuntu and windows xp operating systems. The results of this focused on the performance related metrics like throughput, execution time, CPU usage which are empirically measured setting up a test bench. The results showed that the throughput of
windows xp is far better than Linux and Linux possessed an advantage of hard disk performance using the ext3 file system when compared with windows [65]. Nowadays operating systems includes more than one kernel.

**Jing Chen, Da-Wei Chang, Chung-Ping Young, Guan-Ying Huang, Su-Lin Chu, Chung-Yuan Ke, Shih-Tun Yen and Tsang-Shuo Kuo in 2011** developed a multi-kernel embedded system on PAC embedded multi-core platform. The multi-kernel architecture comes from running one kernel on each processor core in PAC Duo SOC which is composed of one ARM processor core and two PAC DSP cores. A Linux kernel is ported to run on the ARM core and μC/OS-II is ported to run on either or both of the PAC DSP cores [66].

**Zhang Ting, Jiang Li and Li Nan Liu Hong in 2011** developed a five-fingered, multi-sensory and biomimetic prosthetic hand which is detected using DSP mechanism and an USB chip (ATmega32U2). The proposed impedance control approach allows the fingers to behave as virtual springs [67]. The art of building a new boot loader and integrating the old with a new needs some enough skills.

**Liu Wenyi, Zhan Jianhua and Bai Xianmin in 2011** introduced a principle of writing Boot-Loader and introduces the process of building linker command file. This also gave the steps of on-line programming for flash memory. The technology and methods adopted in the design are flexible, stable and reliable which would be practical for the DSP developers [68].

Data recovery tools in operating systems are very useful to users. But **Tsozen Yeh and Weian Cheng in 2012** developed a system that takes multiple check-points of running process to recover data from a crash. When the system is about to crash, the developed system will provide available checkpoints made for that particular program to users. So they can pick a specific checkpoint to re-input legitimate values or various options to continue the execution of the program just crashed. This system was implemented on a Linux Operating System [69].
The boot loader designed in some papers were static (i.e.) it can be used only for a particular application. The remaining papers describes how an operating system is booted from an USB device through BIOS. The problem for booting through BIOS is, many end users will not know how to get into the BIOS to change the boot priority of USB device.

The proposed work will solve the above problems. Here I have developed a dynamic boot loader which is integrated with a portable MS operating system. The designed dynamic loader identifies the USB device and boots the operating system automatically. Also at the same time it configures the network once the operating system is booted. This operating system is designed to be light weight and the users can carry anywhere with their required applications in a Live-USB stick. It saves the user's session in a virtual file system. The next chapter deals with the details about the FAT file system, how it works and why the FAT file system is used in the USB drive for booting the operating system is discussed.