

Brooks-Iyengars Real-Time Sensing Algorithm for Future

Latesh Kumar KJ*

Department of CSE, Siddaganga Institute of Technology, India

Abstract

The article primarily benevolences a two-decade longstanding and most influential Brooks Iyengar's Hybrid algorithm known as Robust Distributed Computing and Sensing Algorithm. The algorithm has established a foundation mark across various real time operating systems, application areas and fault tolerant schemes. The crucial contribution of the algorithm is majorly found in enhancing the features of MINIX real-time operating system, the hybrid architecture and scalability of the algorithm is proficient enough to encounter the unreliable and distributed sensors data using the Byzantine agreement and distributed decision-making process methods. This article emphasises on inclusion and adoption of most persuasive long running Brooks Iyengar's algorithm in MINIX real-time operating system and their recent enhancement of incorporating the fault tolerant schemes. Further, the richness of algorithm has acclaimed by millions of vivid category of users around the globe in their research and tasks. Most significantly the algorithm is beneficial to DARPA, Open Source - Linux, IT Industry - Bae Systems and BBN technologies, Academics – Universities of Maryland, Georgia Tech, Purdue, Clemson and Wisconsin etc., and Research Labs like Penn State Applied Research Lab (ARL), USC/ISI. Additionally, the scalability of algorithm proved to be beneficial to other domains like cyber physical systems, robot merging, high-performance computing, reliability of software and hardware appliance and artificial intelligence systems. In this paper, we attempt to showcase the use cases and real-time deployments of Brooks-Iyengar algorithm in various aspects of physical world. Finally, the influence of algorithm across real-time MINIX operating systems.

Keywords: Applied research lab; Fast convergence algorithm (FCA)

Introduction

Brook Iyengar algorithm

The current internet world consists numerous automated systems that must communicate with dynamic atmospheres. Because these environments are undeterminable, the systems depend on sensors for the delivery of information in order to perform the computation. The Sensors are unenviable interface across computer systems and internet real world for the communication of data.

The smart intelligence development and deployment into these automated control systems is arduous because of limited sensor accuracy, and the noise in data readings recurrently corrupts the accuracy of data. The Brooks-Iyengar algorithm is widely known as Brooks-Iyengar Hybrid algorithm [1,2], this algorithm acclaimed for its betterment in the accuracy of interval measurement engaged by a distributed sensor network.

The key merits of this algorithm is, even with the faulty sensor presence the smart intelligence of sensor network swaps the measured value and precision value at each node with every peer node and performs accuracy in measured range and value for all nodes of the network [3]. The resilient point of algorithm is that it is a fault-tolerant and distributed and it does not malfunction even if some sensors transmit faulty data, because of this key feature it is used as sensor fusion method. Further, accuracy and precision assurance are proved in 2016 [4]. The algorithm Brooks-Iyengar integrates with Byzantine agreement with sensor fusion to control the presence of noise in sensor data.

The algorithm is designed to channel the space between Byzantine fault tolerance and sensor fusion. Further this algorithm is identified as the first algorithm to amalgamate these dissimilar fields [5]. Principally, it syndicates algorithm of Dolev's [6] for an imprecise contract with fast convergence algorithm (FCA) by Mahaney and Schneider's. The core of algorithm pretends processing elements (PEs) as N and t of them are assumed to be faulty and perform malevolently. It accepts both real and unreal values with noise and uncertainty. However,

the output produced by the algorithm is real value with appropriate stipulated accuracy. The algorithm is further customised to resemble crusader's convergence algorithm (CCA) [7], this adoption increases the bandwidth requirement in processing of algorithm. The benefits of algorithm are wide spread across domain like high-performance computing [8], distributed control, software reliability and real time MINIX operating systems.

The use of algorithm is not restricted to specific domains and applications we've cited. In general, all floating-point computations produce inaccuracy and this varies from machine to machine. This hybrid algorithm offers increased scientific consistency on a distributed system encompassing assorted components. This offers a novel method of resistance in rounding and skewing the errors generated by hardware limitations. Today's cloud based software development and customer requirements are inconsistent, the cloud based various services like software, platform, data, network, security and recovery are different in domain but targets produce a common service to customer.

In these environment fault tolerance and accuracy of services must be assured from end to end terms. The Brook-Iyengar's algorithm is useful and effective in these instances by achieving robust and distributed accuracy because of the novel intelligence of algorithm. The cluster computing involves the critical data and service modules that are important systems which demands the additional strength and accurateness.

Regrettably, data, service and security are compromised often between them. Nevertheless, the usage of algorithm increases the by not

*Corresponding author: Latesh Kumar KJ, Department of CSE, Siddaganga Institute of Technology, India, Tel: 9900550607; E-mail: latesh.kj@hotmail.com

Received June 13, 2019; Accepted January 28, 2021 Published January 28 2021

Citation: Kumar KJL (2021) Brooks-Iyengars Real-Time Sensing Algorithm for Future. J Inform Tech Softw Eng.11: 250.

Copyright: © 2021 Kumar KJL. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

sacrificing accuracy with data, service and security. The fault-tolerance mechanism defined in the algorithm is highly beneficial in both active and passive cluster computing in primary and disaster computing sites. With this algorithm, sincerely robust distributed computing applications can be developed and deployed seamlessly. Today's world is Internet of Things (IoT) and cloud based services, sensors are vital part of the IoT systems and cloud readers. The amount of data communication across current dynamic environments are leading to errors, mechanical failures and uncertainties in sensors. To avoid this backup mechanism are plugged but fault-tolerance and accuracy can't be managed. Hence the Brooks-Iyengar's algorithm has lower bound and upper bound, using this technique inaccuracy are dealt smartly and specifically.

The algorithm is not limited to a specific domain neither restricted to set of computing and hence the wide spread smartness of algorithm is encashed from last 20 years in by various researchers, computing labs and university training materials. In illustrating an example, a robust fault tolerant rail door state monitoring systems is developed using the brooks-iyengar sensing algorithm to transportation applications [9], in this paper the author Buke Ao clearly listed the implementations of brooks-iyengar algorithm in variety of redundancy applications by various research studies [10,11], this comprised a program demonstration through the US Defense Advanced Research Projects Agency (DARPA) with BBN using the Sensor Information Technology for the War Fighter (SensIT) program. This group is committed to develop software's for networks of distributed micro sensors and especially in collaborating the signal and information processing and fusion of sensor data. In specific, the algorithm is extensively incorporated to create practical and survivable sensor network applications developed by Penn State University Applied Research Laboratory (PSU/ARL). The key identification is Brooks-Iyengar algorithm has prominently extended seamlessly by connecting the legacy and edge cutting trends of technology variants in software applications and hardware control systems in cloud and non-cloud systems [9].

The major contribution of algorithm is identified with relevance to Linux and Android operating systems effectively. Truly, tons of various software applications and hardware control systems have encapsulated the brooks-iyengar algorithm to offer fault-tolerant fusion data across billions of users accessing the various services through internet and other sources of digital media. Further, algorithm indirectly benefits across to 99% of world's top supercomputers and 89% of smart phones around the globe.

Real Time Minix Operating System

The Real Time Minix operating system is an enhanced version of MINIX operating system, this was originally programmed by scientist Andrew Tanenbaum for teaching operating system on x86 computer system. The research study and implementation by the author Gabriel Wainer changed the MINIX operating system to support RT-processing named it "RT-MINIX" by adopting Brooks-Iyengar algorithm in the areas like Scheduling Algorithms Selection, Scheduled Queues, Real Time Metrics Collection and fault-tolerant systems [12,13].

Before we explain the deep impact of Brooks-Iyengar algorithm on MINIX operating system, we intend to detail about the MINIX operating system. The detailed understanding on MINIX operating system sets a platform for understanding the Real Time MINIX (RT-MINIX) for various applications services and control systems. The MINIX operating systems drivers, user and system specific servers runs on highest level on the miniature kernel architecture, as illustrated in

figure [2]. The SYS and CLOCK are the two major tasks responsible to support the user-mode sections of the operating system at higher levels. Apart from this programming the MMU, CPU, interrupt handling and IPC are the other privileged operations of the MINIX kernel. Just like any other operating system the functionalities like File system (FS), Memory Management (MM), User Management (UM) and Process management (PM) is offered by MINIX. The key and unique feature of MINIX over other operating system is, stealthily this RS server monitors all the device drivers and various servers inside the operating systems at all time and fixes it automatically when any failure is noticed.

All system calls are focused blatantly by system libraries to right server to manage the kernel communication. Let us consider a user requests a process to run an application task, usually a process is initiated by fork () library function when the process manager approves it by verifying with the memory manager on the process slots availability. If any slot available, then process manager instructs the kernel to produce a copy of the process, all these happens transparently without the notice of the user application task. Just like UNIX the MINIX kernel is responsible of managing hardware and device drivers. This involves process scheduling, interrupt management, memory, device I/O and CPU management. The two major core components of kernel space SYS and CLOCK are explained here because in later sections we illustrate how the Brooks-Iyengar algorithm is seminal for the RT-MINIX enhancement. The SYS control is known as system task, this is vital for all kernel mode operations for the device drivers and key heartbeat channel for user segment servers. Any user request to process internally sends a signal to kernel through the library function, each request is passed to SYS. There are various categories based on the SYS management on kernel calls, to copy data between process SYS calls SYS_VIRCOPY and to configure an alarm SYS_SETALARAM etc. Few new systems call defined in the MINIX are listed in Tables 1-4.

The second core object is CLOCK by which kernel manages the process scheduling, timers, cron services, hardware clock and CPU usage. The interrupt handler will initiate a timer moment when a MINIX system is power on, since then each tick is countered using this interrupt timer. In general, the cooperating servers are created by modulating the operating system, the native MINIX operating system allows third party device drivers untrusted code to run and communicate with kernel, the MINIX is smart and manages the spreading of failures. A tight coupling of devices and library functions are created to intact the seamless communication in the low-level kernel operations. In this paper we are describing MINIX operating system in deep to prove how the brooks-iyengar algorithm is influencing the fault tolerant and robust distributed control systems in RT-MINIX. The Figure 1 below illustrates the architecture of the failure-resilient operating system.

The Figure 2 shows how user space in managed by microkernel on file systems and device drivers for higher level user applications and process threads. In this architecture reincarnation server on right side tracks and monitors the device drivers and services all time to power the auto-fixing procedures.

Influence of brooks-iyengar algorithm

Brooks-Iyengar? the name is all over the globe from last two decades, this algorithm is considered to be the all-time best robust

Kernel Call	Purpose
SYS_VIDEVIO	Read or Write a vector of I/O ports
SYS_VIRCOPY	Safe copy between address spaces
SYS_GETINFO	Get a copy of kernel information

Table 1: Privileged SYS Calls to Kernel.

Data Structure	Parameters	Description
<pre> struct rt_globstats { int actperstk; int actapetsk; int misperdln; int misapedln; int totperdln; int totapedln; int gratio; clock_t idletime; }; </pre>	Acrta_petsk, Act_pertsk Mis_perdln, Mis_apedln Tot_petsk, Tot_petsk Gratio Idletime	Period and aperiodic real time tasks, total running tasks. Total missed deadlines Total real-time task scheduled instance Guarantee ratio between deadline and instances Computing Time in Second (clock Tick)

Table 2: Updated MINIX data structure.

OpenMPI Methods	Description
Isend and Ireceive	Non-blocking sends and receives were used to communicate from sensor to sensor. In order to process the data each sensor needs the data from every other sensor in the network. This means there are a worst case of N^2 messages being passed at any given point. Due to this large number, it is best to use non-blocking communications.
Barrier	This acts as a sync step for all sensors. Barrier merely acts as a join for processes in the context of OpenMPI. It is very useful for a simulation as this to stop one process from being a front runner.
Broadcast	Seeing as this is a timed execution program, it is necessary for each sensor to kill itself after a fixed period of time. However, it is possible for one process to keep running if it gets to the check before all the others. To get around this one thread was designated with the responsibility to check the runtime, and then broadcasted the result to all others.

Table 3: OpenMPI methods implemented using Brooks-Iyengar.

S.N	Area/Technology	User
1	Operating System- MINIX	Andres Tanenbaum
2	IT Industry – Software Systems	BBN, BAE Systems, Sense-IT
3	Research Labs	Penn State Applied Research Lab(ARL)USC/ISI
4	Research Communities – RT:MINIX	Pablo J. Rogina and Gabriel Wainer
5	Transportation – Railway	Buke Ao, BYD Company
6	Education- Training/Journal/Conference	Duke, Wisconsin, UCLA, Cornell, Purdue, Georgia Tech, Clemson, Maryland and LSU University
7	Defense- Thale Groups	UK Defense Manufacturer
8	Navy-Software	Maritime Domain Awareness Software
9	Technology	Cyber-physical systems, Data Fusion, Robot Convergence, High-Performance Computing, Artificial Intelligence Systems
10	Open Source	RT Linux, KURT, YARTOS, Spring

Table 4: Brook-Iyengar algorithm live use cases.

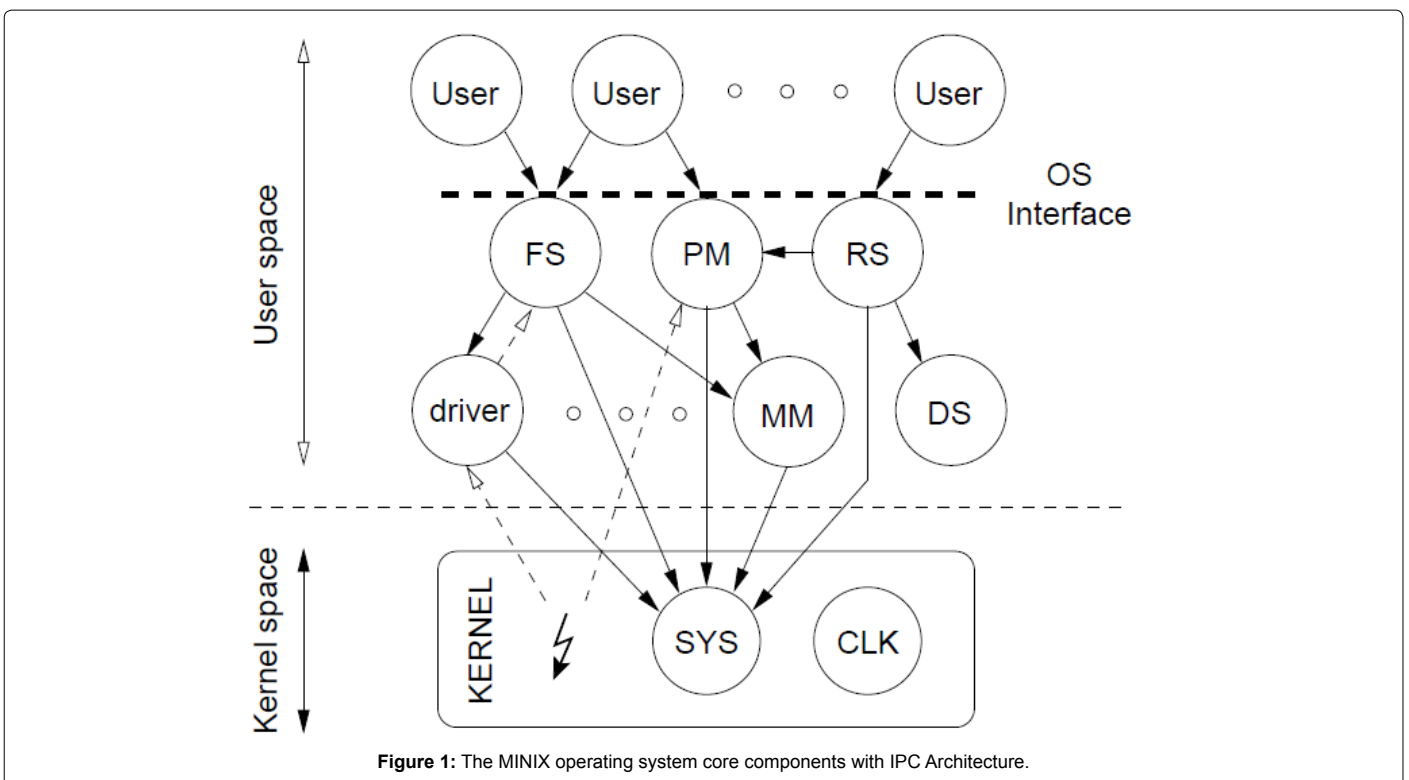
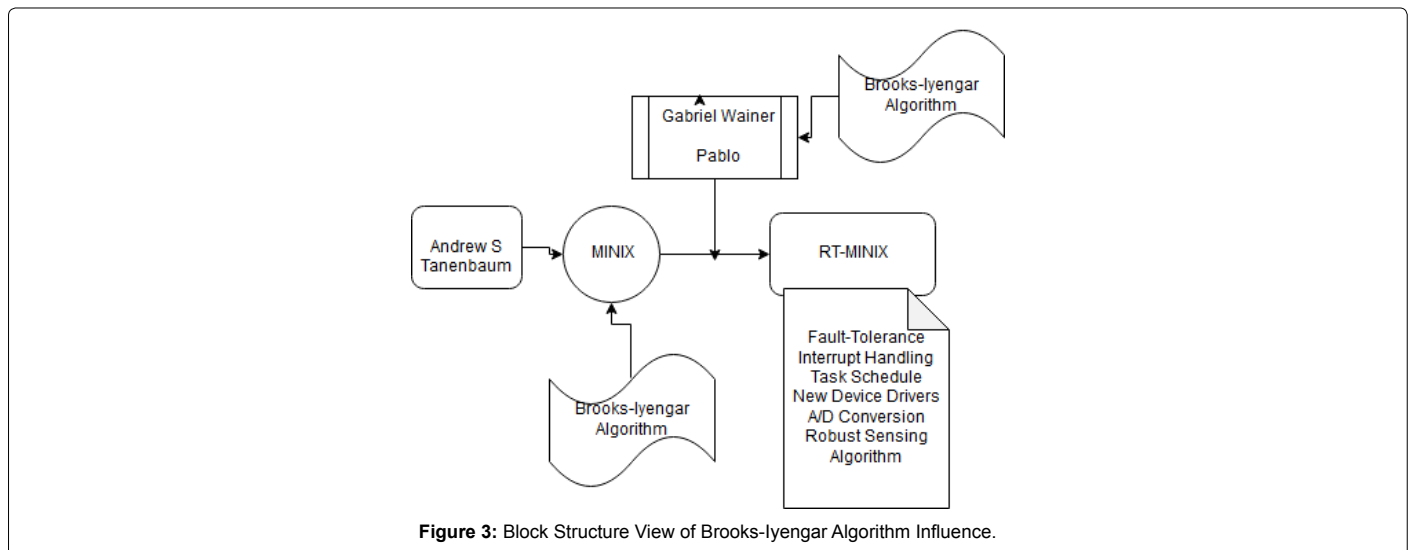
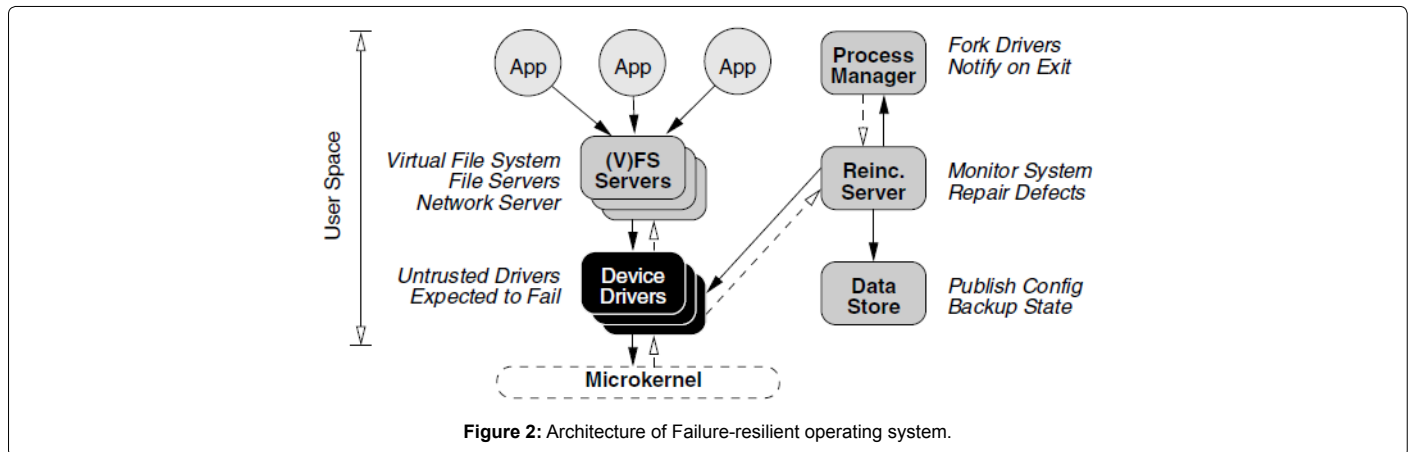


Figure 1: The MINIX operating system core components with IPC Architecture.



algorithm for precision, fault-tolerance and isolation of errors across software applications and hardware control systems.

In this segment we narrate the Brooks-Iyengar algorithm’s influence in various domains like MINIX operating system, sensor networks, software application development, real time extensions, virtualisation, cloud computing and physical cyber systems. To begin with we explain the development and deployment of a distributed sensing algorithm that has major influence on computing systems.

Brooks-Iyengar’s algorithm on MINIX operating system

The Minix operating system powered by Tanenbaum’s was enhanced to Real-Time (RT) Minix operating system and services by Wainer and it is identified as RT-MINIX [12-15]. Further few novel features were added to shape up an academic real time operating system called as MINIX v2. This architecture of design was proposed to train the RTOS with few major topics:

- System Architecture
- Handling Interrupt
- Process Management
- Scheduling of Process
- Fault-Tolerance

- Isolation of Errors

The research study by Gabriel Weiner also mentioned that many other control systems, computer application and real-time systems are created based on the services offered by Brooks-Iyengar algorithm. The services provided by the algorithm on real-time systems, computer applications and various systems are vaguely different from traditional systems. is unique and different from the native operating system.

The Figure 3 describes novel features added to MINIX operating system by Gabriel and Team in creation of RT-MINIX by using the intelligence of Brooks-Iyengar Algorithm. The programming of the MINIX source code was dedicated to provide the real-time controls on various services. Many real-time services were added, to begin with rate-monotonic scheduling [12,16], Earliest Deadline First processor and fault-tolerance are programmed. To make these new changes in the source code of the kernel, the code flow and data structures are slightly modified based on the new updates. Specifically, sensor, timers, schedule and criticality. Further, to adapt live-tasks with interactive CPU bound tasks a multi-queue is developed. The below listed data structure is modified in lieu of RT-MINIX evaluation.

All these changes are tested with various feasibility of MINIX for the real world challenges for real-time development. Numerous work was done using Brooks-Iyengar robust distributed computing algorithm from the testing of novel scheduling procedures to kernel alterations.

In the mean time new version of MINIX was released and hence to sync the RT-MINIX version with MINIX version, some changes were made. The analog to digital conversion, in this update the target was to acquire data from analogic environment as many real-time systems are employed to handle the real process like chemical and a production line [17]. In this requirement the Brooks-Iyengar algorithm's sensor management intelligence is effectivley used for sensing the real world data, to control the noise and to manage the faulty sensors. The interface used for game ports were used to provide the signals from the sensors, this was considered and a device driver for port is developed. The changing environment rely on poor performance of integral systems of RT-MINIX with novel techniques. The Brook-iyengars's algorithm adopted fast convergence algorithm (FCA) to increase the convergence ratio [8].

According the Pablo Ragina and Gabriele Weiner, the algorithm is used extensivley to extend RT-MINIX with posibility of several sensors from a fault-tolerance perception [15]. At the outset, the complete coding was performed based on the all the four algorithm's of hybrid brook-iyengar. The next immediate phase was to integrate the smart capability to make use of the real-time data, to do this four potentiometers were used to sense the signals/data from analogic inputs from the joystick port. These sensor positions are arranged with actual positions for a simulation based robotic arm. An accurate and precise functionality of algorithm was noticed by provinding a exclusive value from the simulated sensors inspite of faulty, at the same time users were offered open chance to modify the data by varying the potentiometers. At last, all the updated code is test for various feasibility and real-time constraints and then the novel algorithm intelligence is united into MINIX kernel. The Figure 4 shows the RT-MINIX Kernel and new feature additions with Brook-Iyengar Algorithm.

The software developers were given a set of functions to work with intellectual sensors, using these it was possible to generate many new services and devices like /dev/js0 and after that smart sensors were able to read data in the presence of faulty sensors. Once the operating system is enhanced with RT services, the demand ascended for various computing tools and applications. The Brook-Iyengar's algorithm needed a test on the novel techniques applied on kernel, in order to evaluate the data structure through vivid system and library calls.

Case Study

OpenMPI+virtualisation

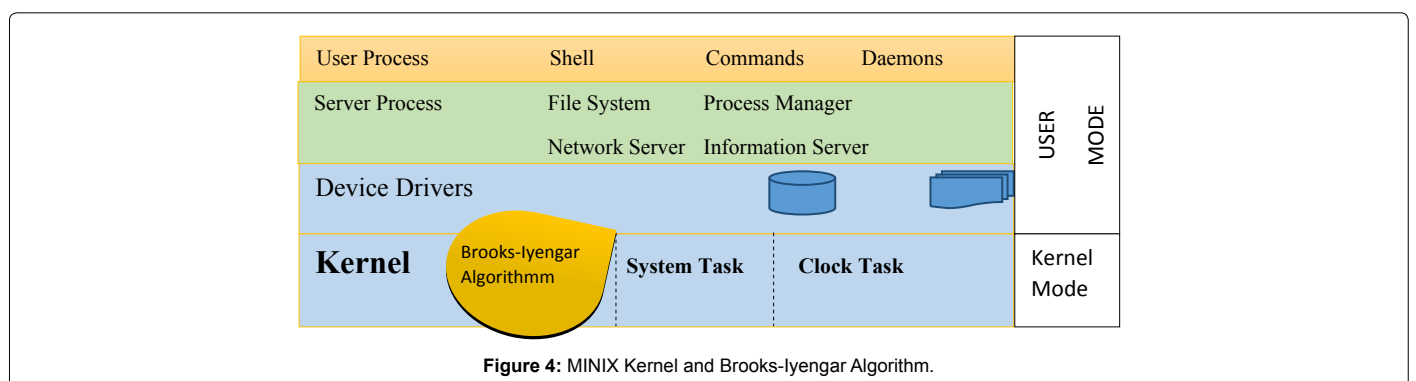
The Brooks-Iyengar algorithm was further implemented on Linux using the OpenMPI [18], this is an open source project created to pass message through interface. This is a collaborative consotium of industry partners, reseach community and groups of academic. Hence, the OpenMPI is powerful and smart because the knowledge, technology and resources are shared from various community.

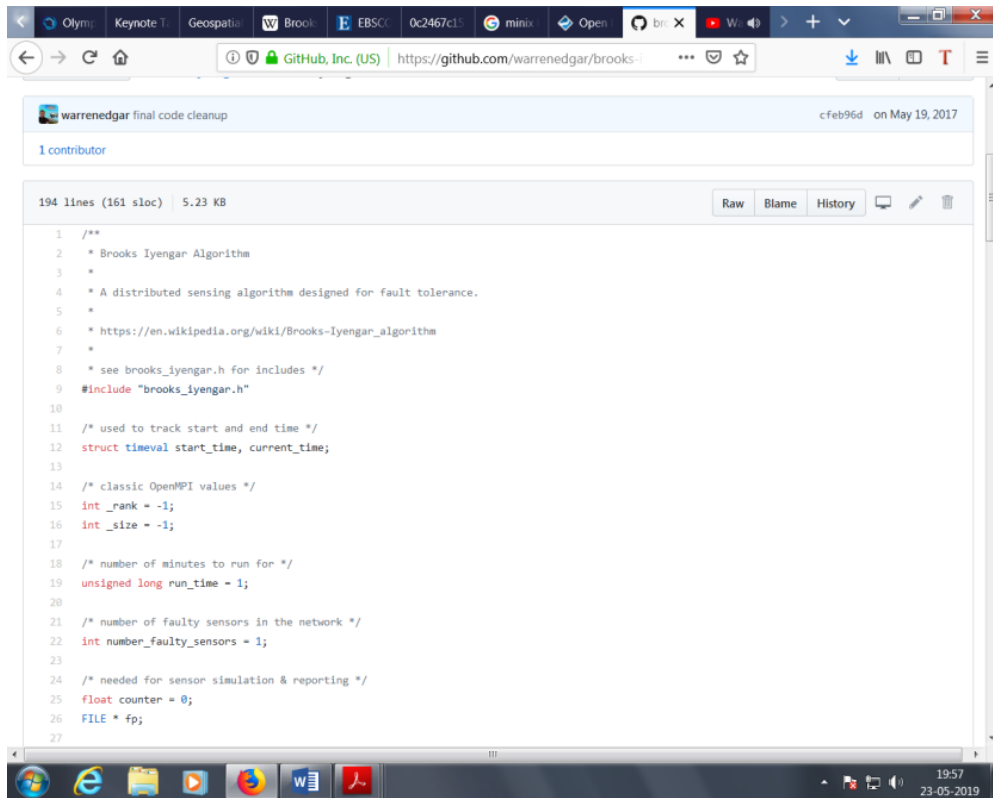
The libraries of MPI provides support to software developers and researchers of computer science and operating researchers. The below Figure 5 shows the hard coded c code in implementing the OpenMPI by using the Brooks-Iyengar algorithm for fault tolerance.

A classical problem in distributed computing is Byzantine Generals' Problem, introduced in the 1982 white paper of the same name. It attempted to formalize a definition for faulty (traitorous) nodes in a cluster and how for the system to mitigate it. Solutions such as majority voting or signed messages were suggested. Majority voting requires that all generals have the same information, a suggestion that isn't always possible. Signed messages are a good to verify that it was the correct node in communication, even if it doesn't verify that the content itself is correct. Both are good suggestions, but it would be more interesting to have an algorithm that can survive a traitorous order every now and then. Enter the Brooks-Iyengar algorithm as an attempt to solve this problem. This algorithm uses sensor fusion to mathematically eliminate the faulty sensors. In short, this is achieved by taking measurements over an interval, the measured interval is then shared between all sensors on the network. The fusion step then happens, by creating a weighted average of the midpoints of all the intervals. At this point you can eliminate any sensors with high variance or use heuristics to choose the reliable nodes. It runs in $O(N\log N)$ time and can handle up to $N/3$ faulty sensors [19-21].

The Figure 6 illustrates the recorded output from OpenMPI implementation for eliminating faulty sensors. The figure depicts three colored output curves ranging from 0 to 6 across the X and Y axis. The analytical results are displayed in blue color, the brooks-iyengar's 100% accuracy in faulty sensor elimination is displayed in green color and the red color curve denotes the discarded lower weights intervals in the last step of pseudocode for eliminating the faulty sensor. The output representation from the figure is included with faulty sensors and overall an average value is considered across all sensors. Because of high activity in the sensor the data represented in the graph is not ideal to visulaise the realistic output of each sensor. Overall, the faulty sensors are controlled from ruining the measurements by benefiting from payback error distribution.

To conclude the obtained results it is better to consider the dumb average because, noise generated from real and faulty sensors are from undeviating distribution. If the algorithm has not performed better then the noise would have been twisted and tremendous in one direction causing the red line curve aggressive over the green line. Overall, the algorithm is very difficult to implement as there were no framework/library and demands precision of coding and adequate infrastructure to achieve best results. The results proved that Brook-Iyengar's algorithm is smart and scalable across various domains like cyber physical system.





```
1 /**
2  * Brooks Iyengar Algorithm
3  *
4  * A distributed sensing algorithm designed for fault tolerance.
5  *
6  * https://en.wikipedia.org/wiki/Brooks-Iyengar_algorithm
7  *
8  * see brooks_iyengar.h for includes */
9  #include "brooks_iyengar.h"
10
11 /* used to track start and end time */
12 struct timeval start_time, current_time;
13
14 /* classic OpenMPI values */
15 int _rank = -1;
16 int _size = -1;
17
18 /* number of minutes to run for */
19 unsigned long run_time = 1;
20
21 /* number of faulty sensors in the network */
22 int number_faulty_sensors = 1;
23
24 /* needed for sensor simulation & reporting */
25 float counter = 0;
26 FILE * fp;
27
```

Figure 5: Open MPI Implementation for Fault-Tolerance using Brook-Iyengar.

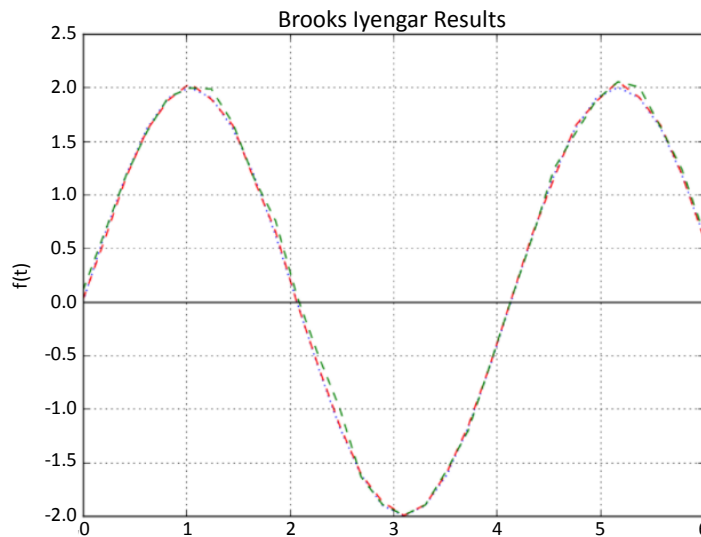
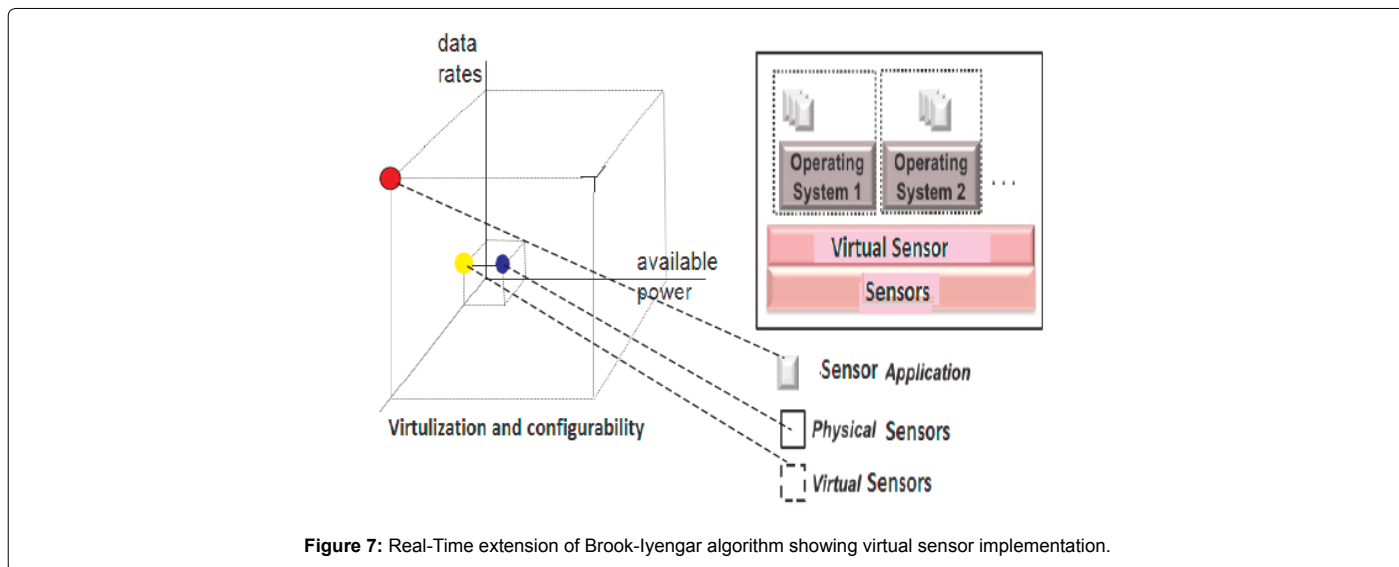


Figure 6: OpenMPI with Brook-Iyengar Fault-Tolerance Results.

Virtualization

The team found that the hardware support fails to provide an unambiguous performance advantage for two primary reasons: first, it offers no support for hardware fault detection; second, it fails to co-exist with existing software techniques for distributed sensor networks. They look ahead for emerging techniques as shown in Figure 7 for addressing this sensor virtualization problem in the context of software-assisted distributed fault-tolerance.

A real-time operating system can be modified to host the abstract fault-tolerant sensor layer with its own dynamic interval estimator, which is a mapping of the real-sensors. An Abstract Sensor is a sensor that reads a physical parameter and gives out an abstract interval-estimate I, which is bounded and connected subset of the real line R. A Correct Sensor is an abstract sensor where the interval estimate contains the actual value of the parameter being measured. If the interval estimate does not contain the actual value of the parameter being measured, it is called Faulty sensor.



Conclusion

In this article the acceleration, effectiveness and liveness of two decade old Brooks-Iyengar Algorithm is illustrated. Since today's technology does not guarantee success and safety in all situations, the Brooks-Iyengar algorithm can significantly improve the fault tolerance of systems by providing a greater margin of safety for operations. This algorithm provides the robust implementation and seamless scalability under faulty sensor conditions for various domains. Finally, the algorithm "Stand The Test of Times" from last two decades and hope it continues the successful journey further.

References

- Dolev D (1982) The Byzantine Generals Strike Again. *J Algorithms* 3: 14-30.
- Richard Brooks R, Sithrama Iyengar S (1996) Robust Distributed Computing and Sensing Algorithm. *Computer* 29: 53-60.
- Ilyas M, Mahgoub I (2004) Handbook of sensor networks: compact wireless and wired sensing systems (PDF). CRC Press, p: 672.
- Buke A, Yongcai W, Lu Y, Richard RB, Iyengar SS (2016) On Precision Bound of Distributed Fault-Tolerant Sensor Fusion Algorithms. *ACM Computer Surv* 49: 5: 1-5.
- Dolev D (Jan 1982) The Byzantine Generals Strike Again. *J Algorithms* 3: 14-30.
- Lamport L, Shostak R, Pease M (1982) The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems* 4: 382-401.
- Dolev D, Nancy AI, Shlomit SP, Eugene WS, William EW (1986) Reaching Approximate Agreement in the Presence of Faults. *Journal of the ACM* 33: 499-516.
- Mahaney S, Schneider F (1985) Inexact Agreement: Accuracy, Precision, and Graceful Degradation. *Proc Fourth ACM Symp Principles of Distributed Computing*, pp: 237-249.
- Buke AO (2015) Robust Fault Tolerant Rail Door State Monitoring Systems: Applying the Brooks-Iyengar Sensing Algorithm to Transportation Applications. *International Journal of Next-Generation Computing* 8: 108-114.
- Kumar V (2012) Computational and compressed sensing optimizations for information processing in sensor network. *International Journal of Next-Generation Computing* 3: 328-332.
- Ao B, Wang Y, Yu L, Brooks RR, Iyengar SS (2016) On precision bound of distributed fault-tolerant sensor fusion algorithms. *ACM Compute Surv* 49: 1-5.
- Pablo JR, Gabriel W (1999) New Real-Time Extensions to the MINIX operating system. *Proc of 5th International Conference on Information System Analysis and Synthesis (IASS'99)*.
- Gabriel AW (1995) Implementing Real-Time services in MINIX. *ACM Operating System Review* 29: 75-84.
- Tanenbaum Andrew S, Woodhull Albert S (1999) *Sistemas operativos: Diseño e Implementacion 2da Edicion*, ISBN 9701701658, Editorial Prentice Hall.
- Brooks R, Iyengar S (1996) Robust Distributed Computing and Sensing Algorithm. *IEEE Computer* 29: 53-60.
- Chakrabarty KI, Qi SSH, Cho EC (2002) Grid Coverage of Surveillance and Target Location in Distributed Sensor Networks, *IEEE Transactions on Computers* 51: 1448-1453.
- Krishnamachari B, Iyengar S (2004) Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks. *IEEE Tran Comp* 53: 241-250.
- Warrenedgar (2019) An implementation of the Brooks-Iyengar algorithm using OpenMPI.
- Penn State University (2013) Reactive Sensor Networks, AFRL-IF-RS-TR-2003-245, Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS), Defense Advanced Research Laboratory.
- Junkil P, Radoslav Ivanov P, James W, Miroslav Pajic Sang HS, Insup L (2017) Security of Cyber-Physical Systems in the Presence of Transient Sensor Faults. *Journal ACM Transactions on Cyber-Physical Systems*.
- Kumar V (2013) Impact of brooks-Iyengar distributed sensing algorithm on real-time systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, p: 1.