# An Architecture for Dependable Connectivity in OSGi-Enabled Dynamic Distributed Systems

Ali Raza
School of Computing and Communication
Lancaster University
Lancaster, UK
a.raza15@lancaster.ac.uk

Keivan Navaie
School of Computing and Communication
Lancaster University
Lancaster, UK
k.navaie@lancaster.ac.uk

Richard Nicholson
CEO Paremus
London, UK
richard.nicholson@paremus.com

***Abstract* — From air pollution monitoring to debatable surveillance for better security, dynamic distributed systems led to the birth of versatile smart environments. Dependability of such systems is challenged by the reliability of the communication links between various sub-systems. In this paper, we address this issue by designing a TCP/IP based Client-Server architecture using diverse channels, to ensure zero tolerance with regards to outage of service. The prototype system uses Raspberry Pi 3, as a remote client, to intelligently communicate with the server node by choosing one or a set of available communication links, e.g., Ethernet (LAN), Wireless-LAN (Wi-Fi), and Bluetooth channels. We further implement the system using Open Services Gateway Initiative (OSGi), a modular and interpolate-able code foundation, to withstand the challenges faced by the modern software industry e.g. complexity and scalability. Prototype system was successfully tested for hardware and software fault tolerance using different test scenarios to ensure uninterrupted service delivery. In the end, we also present a machine learning technique to mitigate the effects of severe channel hostilities for diverse channels system. The results show improvement in the quality of data transmission by exploiting the flexibility of alternate channels. We demonstrate this intelligent and seamless communication link switching technique using Support Vector Machine (SVM) in MATLAB.**

***Index Terms*—Dynamic distributed systems, dependability, Support Vector Machine (SVM), reliability, security, software complexity, modular programing**

## I. Introduction

Highly dynamic distributed systems such as Internet of Things (IoTs) [1] and Cyber Physical Systems (CPSs) [2] will elevate the way of living in the developed world [3]. The most imminent challenge faced in the commercialization of these technologies is the reliability and evolvability of such systems. Heterogeneous devices including sensors, actuators and computing nodes connected by the CPSs must qualify the strict requirements on latency, range, throughput and high levels of security, which is crucial for the success of applications like personal healthcare and autonomous driving [4]. Similarly, legacy systems for industrial automation can't adapt to the changing conditions and emerging technology needs. Hence, they are becoming inadequate due to strong pressures of cost, quality and customizations of products in highly flexible and responsive production systems [5]. The market and business evolution are pushing modern systems to be more flexible and scalable to handle agile fluctuations with real-time reactivity. These considerations lead this work to investigate how strong modularity and reliability techniques could be combined to realize dynamic, robust and evolve-able distributed systems.

Distributed systems are susceptible to faults. Since the early 90s, significant progress has been made to make the system immune to hardware faults. However, resilience in the application layer was often ignored. Fault tolerant application design is equally important because even if a system is immune to network break-down, it may still collapse due to the faults in the software architecture [6]. It is also important to differentiate between failure avoidance and recovery characteristics. In a complex software system, automated and rapid recovery of low level service allow higher level service to appear to be "fault-tolerant". Research by Berkley/Stanford Recovery Oriented Computing (ROC) group [7] demonstrated that rapid recovery from failure is more important that fewer failures but with longer recovery times [8]. Now, with a multi-fold increase in complexity of embedded software, this problem has exacerbated more than ever before. Therefore, by focusing on Mean Time To Recovery (MTTR) instead of Mean Time To Failure (MTTF) higher availability of CPSs application could be provided and the dependability metric for such systems should swivel around both hardware and software fault tolerant designs. In this paper, our focus is on dependability of communication among sub-systems along with modular application design for dynamic distributed devices.

## II. Survey of Previous Work

The term "dependability" according to the Telecom Glossary of American National Standard Institute (ANSI) is interpreted as "*the ability to deliver service that can justifiably be trusted*". Public service applications such as ISDN and ATM provide a high level of network reliability. In such systems, network failure is avoided using advanced redundancy techniques [9]. It is also shown that such systems are also tolerant to software faults which guarantees strict service availability to meet the requirements of telecommunication standards.

Similarly, a heterogeneous network of several interconnected devices, mobile objects and the cloud forms a Mobile-Cyber Physical System (M-CPS) [10, Fig. 1]. Authors in [10] states that "*the foremost requirement for M-CPSs is its reliability*". They further interpret dependability as availability, reliability, safety, integrity, maintainability of the system.

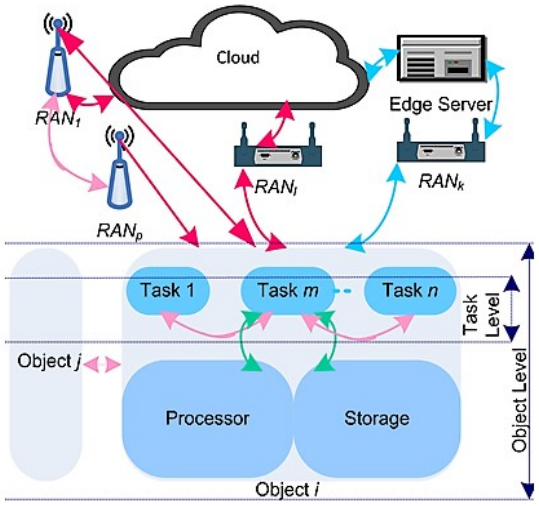


Fig. 1. An overview of Mobile Cyber-Physical Systems (MCPSs) [10].

Latest research argues that dependability will play a major role towards the wide acceptance of M-CPSs. Primarily, the argument is based on highly dependable system operation limited by reliable connectivity. The author of [11] proposes a wireless bus for robust the connectivity of M-CPSs. Therefore, the widespread acceptance of technology is highly dependent on the ability to deliver promised services reliably.

Open Services Gateway Initiative (OSGi) is the open industry standard for modularity. It provides an underpinning to build and host next generation lightweight and adaptive applications for ongoing operation and maintenance. OSGi directly addresses the durability challenge identified by Defense Advanced Research Project Agency (DARPA) [12]. In a software engineering research proposal, DARPA says, "*Modern-day software systems, even those that presumably function correctly, have a useful and effective shelf life orders of magnitude less than other engineering artefacts… While an application's lifetime typically cannot be predicted with any degree of accuracy, it is likely to be strongly inversely correlated with the rate and magnitude of change of the ecosystem in which it executes*". Using OSGi, applications can stop, restart, update and change unexpectedly at the runtime. Therefore, modularity through OSGi specifications provides the foundations for adaptable, interoperable, evolvable platform for Smart Cities and Industry 4.0.

In today's software industry, complexity is one of the most perpetuating problems [13]. Modularity in software design restricts the complexity of software, and hence keeps it manageable to operate and/or upgrade. Guardian reported, "*Digital infrastructure exceeding limits of human control, industry experts warn*" [14].

OSGi services are abundantly found in smart homes. Researchers at Austrian Institute of Technology (AIT) exploits modularity and flexibility of OSGi platform to facilitate independent living of elderly people. In [15], the potential of OSGi services for in-home networking is unveiled for interconnection of different home appliances. It is further shown that OSGi services can be used with multiple heterogeneous technologies because OSGi services can be used to provide application programming interface (API), not just the underlying implementation!

This work is intended to show dependable connectivity using OSGi enabled modular and scalable programming, to cater both dimensions of the identified problem, i.e., hardware and software faults, in distributed systems milieu. Rest of this paper is organized as the following. We present the system Architecture and prototype design in Section III. Then the test results are provided in Section IV, followed by intelligent switching technique in Section V. Finally, we conclude our work in Section VI.

## III. Architecture and Prototype Design

### *A. Software Architecture*

A Client-Server architecture for distributed systems has numerous applications and it is common in the related literature. Authors of [16] presents a client-server model for the wireless connectivity of mobile robots. They utilize this model for distributed mobile robots to work autonomously. The leader robot transmits the sensed information to server where most of the processing and decisioning of path planning takes place. Fig. 2 presents the architecture of OSGi Client-Server system using redundant channels. Rather than a single communication channel - the idea of the project was to introduce diverse/ optionality via selection from a few potential channels. The client then initiates a connection to the server using a specific port number and most robust communication channel for security and reliability, respectively.

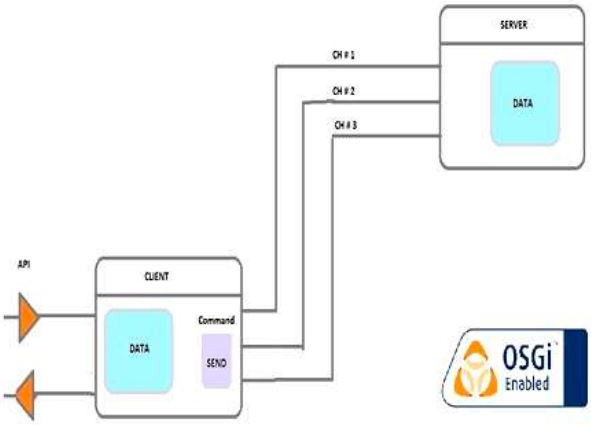


Fig. 2. OSGi based client-server architecture [23]

The architecture is designed to use client as an information transmitting terminal which may be connected to a sensing device. The client has an *Application Programming Interface* (API) and a set of *Commands* which supports the transmission process in the application design. The sensed information is sent to the server using diverse channels. The algorithm attempts to use Ethernet, Wi-Fi and Bluetooth in accordance to the robustness of each channel. In the available channels, Ethernet can provide the highest *data rate* and lowest *Bit Error Rate* (BER), hence it is regarded as primary channel of

communication. In case of node failure, alternative channels (Wi-Fi and Bluetooth) are tried in order.

Later, to interpolate OSGi applications, the client and server systems were embedded with Automatic Repeat Request (ARQ) algorithms [20]. ARQ protocols helps to mitigate the effects of channel distortions. We used *Selective Repeat Protocol* (SRP) [20] to efficiently utilize the bandwidth of active channel of communication. The SRP helps to selectively re-transmit a packet of data that is corrupted due to channel hostility.

### *B. Prototype Design*

Raspberry Pi is a powerful yet inexpensive device, and is being used in communication, safety, weather sensing and energy-savings in IoTs environment. In [18] Raspberry Pi is used as Web node sensor for IoTs application in home automation. Authors of [19] shares an exhaustive list of applications areas of Raspberry Pi as a prototype in sensor networks. In recommendations, it is proposed that Pi can also be used in e-health and education besides sensors networks. As depicted in the prototype design, Fig. 3, Raspberry Pi 3 is used as remote client system to communicate with a computing system acting as a server system using an OSGi enabled platform.

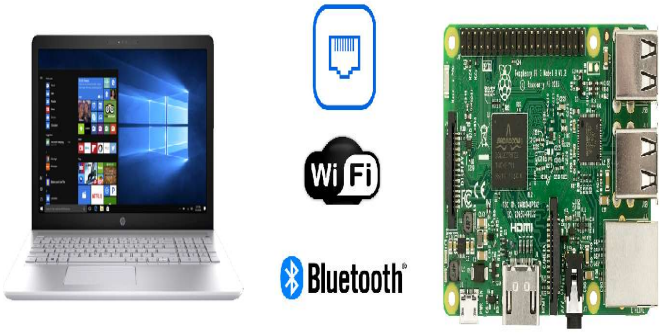


Fig. 3. Hardware system testing prototype

Private networks for Ethernet, Wi-Fi and Bluetooth connectivity were deployed for security in comunication. By enabling internet sharing in the laptop (Fig. 3), a Etherent conection was provided to the Raspberry Pi using RJ-45 cable. Mobile hotspot in laptop was used to provide IEEE 802.11n supported Wi-Fi service. Energy efficient bluetooth radio v4.0 was used for Bluetooth pairing between the devices.

Further, the two terminals requires OSGi client and server applications. JAR files were remotely prepared and transferred to respective terminals. In our OSGi application design, each connection could be identified using a unique IP address. At the client terminal, the API has two fields with the following syntax:

**send *<IP Address> <Data>***

'send' is identified as a transmission command by the client terminal, whereas two API fields contains the *IP address* of a connection and the transmitting *data*, respectively. Complete stages of prototype development are summerised in Fig. 4.

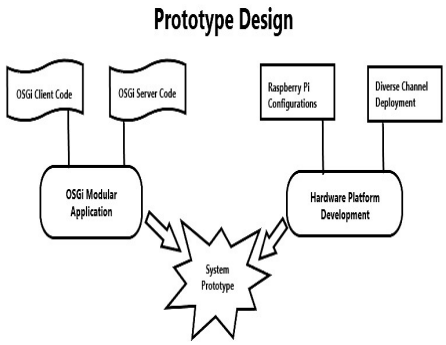


Fig. 4. Stages in prototype development

## IV. Prototype Testing and Results

A distributed system is fully immune to hardware faults if the events of hardware failure are mutually exclusive for different communication nodes. According to [9], all the communication nodes should have independent failure modes for reliable communication. Our OSGi based hardware fault tolerant system is designed to exhibit independent failure events in alternate channels. Moreover, modular software approach provides robustness at the application layer. System is tested in the following scenarios:

### *A. Case One*

Each communication node was independently tested to serve as a medium of communication between client and server terminals. Firstly, to ping the server system, test data was transmitted from the client using each of the three channels separately to ensure all the three channels were properly set. Later, after integrating all the redundant channels in the system, arbitrary data was transmitted using Ethernet, and an acknowledgement (ACK) was received from the server for the sake of confirmation (Stop-and-Wait ARQ). To emulate the failure of a communication node, Ethernet cable was suddenly unplugged. With an attempt to use a faulty node, an error was thrown with network latency of 10 ms. In-spite of network breakdown, uninterrupted transmission could be continued using Wi-Fi by the virtue of quickly recoverable and fault-tolerant OSGi applications.

When a network failure was emulated in the Wi-Fi network, on average the network latency was 2 seconds until another channel could be used to continue transmission. Similarly, successful testing was carried out using different combinations of faultless and faulty channels, to ensure that each node works during a failure event, independent of the other, along with successful toggling between alternate channels. Modular programming successfully enabled the application to quickly recover from unexpected network disruptions. Therefore, it prevents the system failure at the application layer while migrating from one communication node to another. Hence, obtained results serve as an evidence of software and hardware fault tolerance in the proposed model.

### *B. Case Two*

Application layer protocols were used to maintain the integrity of transmitted data i.e. to withstand the effects of

channel hostility. Selective Repeat Protocol (SRP) was adopted to due to performance reasons. Research shows that, SRP enables efficient utilization of the most expansive resource in the communication system – the bandwidth [20]. Using a large window size, high data rates can be achieved because only corrupted data packets are retransmitted, not the entire window like in Go-Back-N. SRP is also desired because merely Transport layer protocol is insufficient to guarantee the quality of service in the wireless transmission of data. Therefore, the application layer protocol was embedded and tested into the system along with TCP/IP to provide better data rates and guarantee Quality of Service (*QoS)* requirements in the system.

OSGi client and server applications were interpolated to include SRP algorithm to investigate the scalability of application designs. SRP was coded using an arbitrary window size to test selective re-transmission for a corrupted packet. Packet corruption was emulated using a software generated distortion. As a result, non-acknowledgement (NAK) messages were returned by the server. The client system re-transmits the NAKed packets selectively and continues transmission until the end of data. In the diverse channels scenario, if the same packet is NAKed twice, it could be sent using an alternate channel. Hence, in modular application design, use of SRP proves the scalability of OSGi applications without increasing the complexity. Table 1 summarizes the results in Section IV.

TABLE I. TABLE OF RESULTS

| **Test Case** | **Results** | |
|---|---|---|
| | ***Hardware Design*** | ***Software Design*** |
| ***Case One*** *(Hardware Failure test)* | Immunity to communication node failure (10ms to 2s network latency) | Unexpected faults tolerant software in the event of network breakdown |
| ***Case Two*** *(Scalability test)* | Alternate channels offer flexibility against data corruption | Meets the complexity and durability challenges with interpolate-able app design |

## V. INTELLIGENT SWITCHING USING SUPPORT VECTOR MACHINE (SVM)

To mitigate the effects of severe channel hostilities, we further propose a machine learning technique to intelligently use redundant connections for reliable and superior services delivery. Using this technique, we try to determine if a channel is severely noisy by inspecting the sequences of ACKs or NAKs returned for the transmitted data. When the channel conditions are bad for data, the receiver returns more or consecutive sequences NAKs due to packet corruptions. As a result, by learning from the pattern of NAKs, autonomous switching of a communicating terminal can be triggered using machine learning techniques.

The effect of increasing channel hostility can be observed in the form of increasing number of NAKs returned for a transmitted sequence. So, extracting the information about number of NAKs in a frame is used to define so called tolerable NAK frequency. Then, we develop a Support Vector Machine (*SVM*) based binary classifier to learn and detect if NAK frequency is within a tolerable range in an observed frame. If the SVM algorithm classifies a frame as *abnormal*, having intolerable NAK frequency due to channel adversities, then the process of connection change is initiated using this detection. Subsequently, transmission could be continued on an alternate channel. The proposed technique is independent of connectivity at physical layer which also makes it suitable for deployment in diverse connections systems. Fig. 5 summarizes the stages of development.

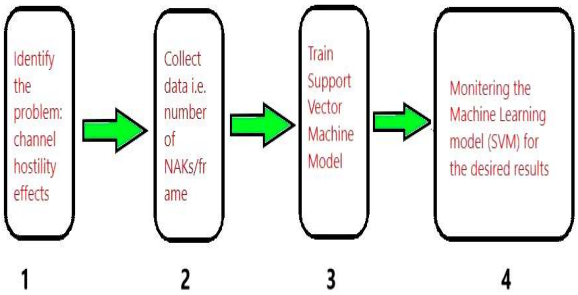


Fig. 5: Stages to implement machine learning technique

### *A. Support Vector Machine (SVM)*

After the recent progresses in statistical learning, Support Vector Machine (SVM) emerged as a new generation learning technique. SVM provides state-of-the-art performance for puzzling applications like hand-written character recognition, image classification, text categorization and bio-sequences analysis [21].

SVM can be applied when data has two classes. It works by finding the optimal hyperplane which completely separates the data points of one class from the other. The optimal hyperplane is the one that exploits largest margin between two classes. Here this margin is defined by maximum width of the slab parallel to the hyperplane, without any in-between points (Fig. 6).

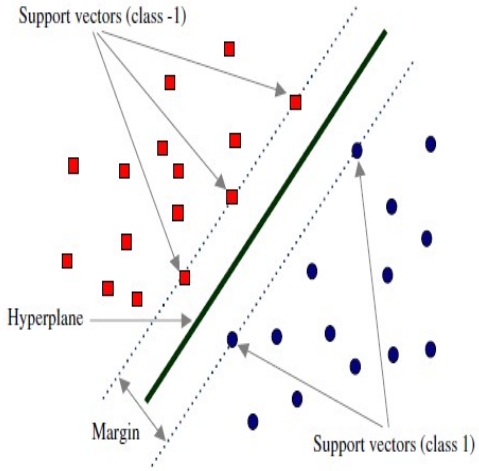


Fig. 6: Two class SVM classifier [24]

In the above figure, support vectors are the data points that are closest to the separating plane, as pointed on the boundaries of the slab. The circular points are marked under class +1 whereas rectangular ones indicate the class -1.

Suppose that for training, a set of points $x_j$ are mapped to its' respective class $y_j$. Assuming that $d$ is used to represent dimension of the data under consideration, then $x_j \varepsilon R^d$ and $y$= +/- 1. As a result, the equation of the hyperplane can be written as [22]:

$$f(x) = x'\beta + b = 0 \quad (1)$$

where $\beta \varepsilon R^d$ and $b$ is a scalar

To find the optimal hyperplane, $||\beta||$ must be minimized using $\beta$ and $b$ such that for each data point $(x_j, y_j)$

$$y_j f(x_j) \geq 1 \quad (2)$$

The support vectors indicated on the boundary of hyperplane (Fig. 6) are those $x_j$ for which

$$y_j f(x_j) = 1 \quad (3)$$

For simplification, the problem can be reduced to minimizing $||\beta||$ such that optimal solution $(\beta^+, b^+)$ enables the classification of vector $w$ as

$$\text{class}(w) = \text{sign}(w'\beta^+ + b^+) = \text{sign}(f^+(w)) \quad (4)$$

where $f^+(w)$ is the classification score representing the distance $w$ from the decision boundary.

### *B. Data Engineering*

We develop our SVM model using MALTAB. A set of data comprising ACKs and NAKs was collected by carrying out a simulation of data transmission over a noisy channel. As a result, a sequence of 10000 ACKs and NAKs was used to train the SVM model. One frame is defined using 10 such consecutive sequences. Lastly, we extract the useful information i.e. number of NAKs in each sequence. To overcome the effect of burst errors, we analyze the collected data to determine a limit for tolerable number of NAKs in each sequence. The following plot shows per frame weight of NAKs in our collected data.

### *C. Training the Model*

After analysis, we defined a classification criterion for SVM. In our algorithm, to reliably deliver the promised services, NAK frequency must be maintained below 3. So, we used SVM to detect 3 or more NAKs in a sequence and classify such a frame under class -1. After such a detection, the process of switching a connection, to use an alternate channel, can be initiated by the system.

We train our SVM model using Linear Kernel Function. Fig. 8 shows the position of support vectors at the boundaries of two class levels (+1 and -1). A frame having tolerable NAK frequency is mapped at +1 whereas a sequence with 3 or more NAKs is identified as -1 by the trained model. The latter will indicate the effect of changing channel conditions.

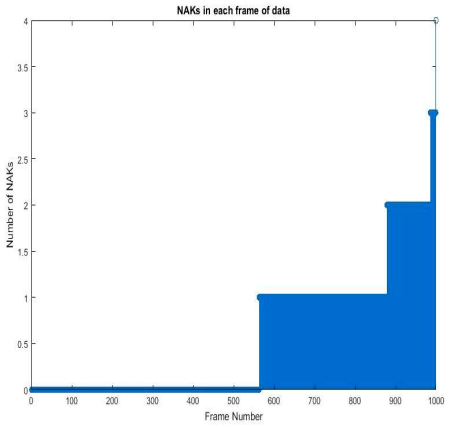


Fig. 7: Number of NAKs in each frame of training data

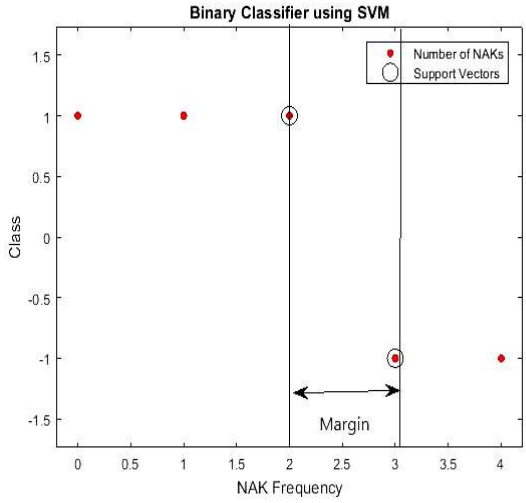


Fig. 8: SVM classifier after training

### *D. Deployment Scenarios and Monitoring*

To demonstrate the results, we deploy SVM model in a simulated environment. A diverse channels system is developed using MATLAB. We developed two Additive White Gaussian Noise (AWGN) based noisy channels - Channel A and B. It is assumed that instantaneous conditions in two channels are not necessarily the same. A binary sequence of a 1000 packet to be transmitted over channels A and B is defined. To mitigate the effects of severe channel hostilities and observe the improvement in transmission after deploying SVM, following test scenarios were prepared:

*1) Transmit all the data on Channel A without SVM.*

*2) Initiate the transmission process on Channel A, and deploy trained SVM model at the transmitter. SVM model inspects each frame of acknowledgements returned by the receiver. As soon as number of NAKs exceed the tolerable range in an examined frame, it is flagged by the model. Then, the process of connection change is initiated. Further data is carried on Channel B.*

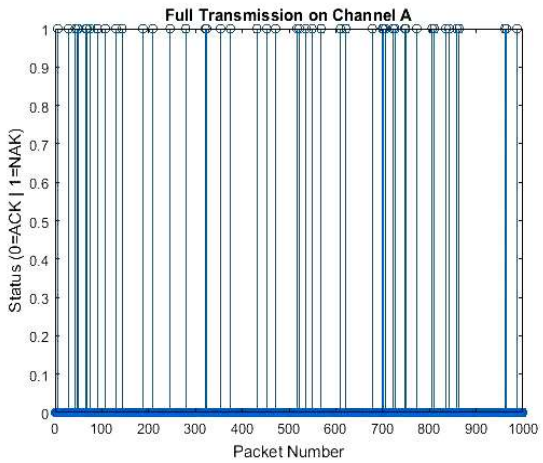


Fig. 9: Pattern of ACKs/NAKs on Channel A

Based on the above criteria, simulation results are discussed between Fig. 9 to 13.

Fig. 9 shows the pattern of ACKs and NAKs when all data was transmitted on Channel A without harnessing the advantages of machine learning. The vertical lines show the erroneous receptions resulting in NAKs caused by channel distortion. Consecutive sequence of NAKs (condensed vertical lines) indicate burst errors due to channel conditions. This implies that more re-transmissions must be done to successfully deliver all the packets to the receiver. This is not desirable in highly reliable systems because propagation delays would undermine the *QoS*.

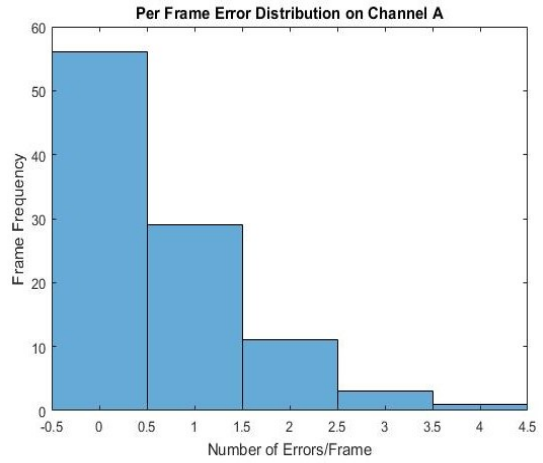


Fig. 10: NAKs frequency per frame on Channel A

The histogram of number of errors observed in each frame on Channel A is depicted in Fig. 10. Burst errors due to channel disturbances results in 3 or more errors in a small proportion of inspected frames.

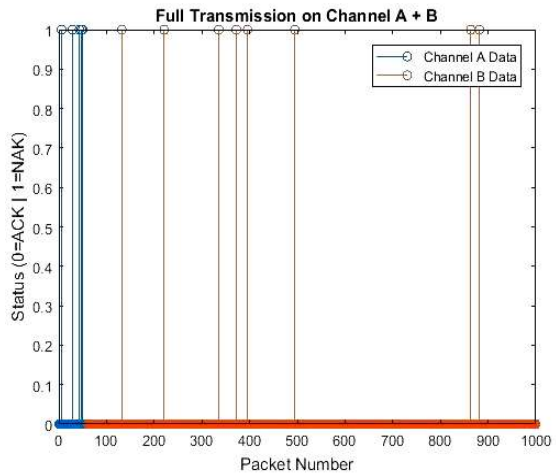


Fig. 11: Pattern of ACKs/NAKs on Channel A and B

Fig. 11 shows the behavior of Channel B towards the same data. As the system detected the first burst of errors due to channel harshness, connection change was triggered using SVM. As a result, data transmission was shifted to the alternate channel. Simulation results showed marked improvement in reducing the number of errors (fewer and spaced vertical lines) and hence the number of re-transmissions on Channel B. Therefore, SVM model helps to predict the severe channel hostilities by inspecting the weight of NAKs.

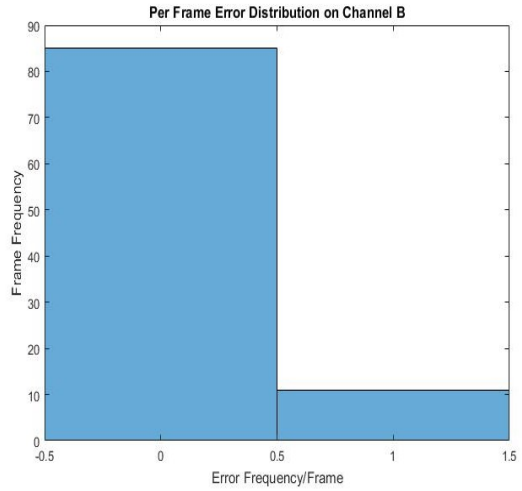


Fig. 12: Per frame NAKs frequency on Channel B

The histogram in Fig. 12 shows that there was a maximum of 1 error in 10 frames whereas majority of frames were reported error free. This clearly demonstrates the advantages of intelligent channel switching using SVM model. It enabled efficient utilization of redundant resources leading to smooth transmission of data after switching onto Channel B.

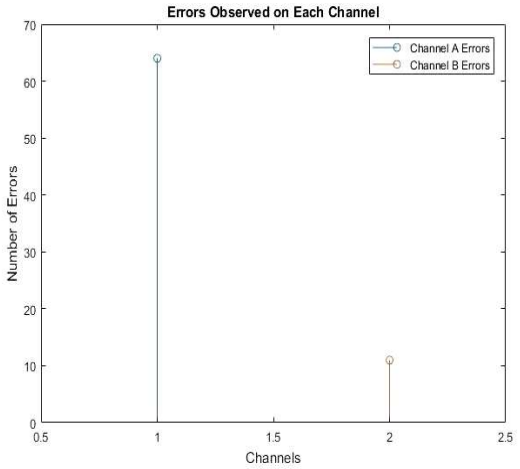


Fig. 13: NAKs frequency comparison on Channel A and B

Fig. 13 is a comparative graph of number of errors occurred on each channel. Channel A corrupted 64 packets whereas only 12 packets were corrupted on Channel B after SVM detected the first burst of errors on Channel A. Hence, re-transmissions rate on Channel A is at-least 4 times higher than that of Channel B.

## VI. Conclusions and Recommendations

Dependable connectivity is a growing notion in academic research. Soon, this idea will draw more attention of the researchers associated with M-CPSs and IoTs. This concept is the spine of a system which has zero tolerance regarding the outage of service. The adopted approach is unique and different because it combines dependable connectivity along with interpolate-able OSGi technology which allows quick recovery under unforeseen circumstances and reduce the pain and cost of software maintenance by mitigating the complexity hazards of multi-purpose commercial applications. The prototype design of distributed system is both integrate-able and scalable in size, geography and administrative dimensions. Many heterogeneous devices can be brought together using the presented architecture in OSGi environment. Choice of different channels could be crucial depending upon application areas. In modern cities, Ethernet and Wi-Fi would connects most of the devices whereas Bluetooth may be preferable in energy-constrained environments.

SVM is the most efficient machine learning method and confers many advantages to our system. It can be used to model and adapt against channel specific error patterns or severe channel hostilities and develop appropriate algorithms. Consequently, it would also reduce the receiver complexity as less repeats transmissions would optimize sorting and computational processes. Hence, SVM will enhance reliable delivery of services by intelligently and seamlessly switching connections.

## Acknowledgment

The authors would like to thank Paremus Ltd UK for arranging OSGi training course at London. We are also grateful to anonymous reviewers for their comments to improve this works. This work was partly supported by the EU Horizon 2020 project no. 690750-ATOM-H2020-MSCA-RISE-2015 and partly by the EPSRC IAA in Lancaster University.

## References

[1] P. Ray, "A Survey on Internet of Things Architectures", EAI Endorsed Transactions on Internet of Things, vol. 2, no. 5, p. 151714, 2016.

[2] X. Hu, J. Cheng, X. Li, W. Tan, Q. Liu and Z. Sheng, "Mobile Cyber-Physical System", Mobile Information Systems, vol. 2017, pp. 1-2, 2017.

[3] J. Shi, J. Wan, H. Yan and H. Suo, "A survey of Cyber-Physical Systems," 2011 International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, 2011, pp. 1-6. doi: 10.1109/WCSP.2011.6096958

[4] A. Burg, A. Chattopadhyay and K. Lam, "Wireless Communication and Security Issues for Cyber–Physical Systems and the Internet-of-Things", Proceedings of the IEEE, vol. 106, no. 1, pp. 38-60, 2018.

[5] P. Leitão, A. Colombo and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges", Computers in Industry, vol. 81, pp. 11-25, 2016.

[6] A.D. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, N. Treuhaft. Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. *UC Berkeley Computer Science Technical Report UCB//CSD-02-1175*, March 15, 2002.

[7] "The UC Berkeley/Stanford Recovery-Oriented Computing (ROC) Project", Roc.cs.berkeley.edu, 2018. [Online]. Available: http://roc.cs.berkeley.edu/. [Accessed: 13- Jun- 2018].

[8] E. Rathgeb, Dependable Communication –vision or illusion. IEEE Proceeding on Software Engineering and Advanced Applications, 2006.

[9] K. Navaie and H. Aghvami, "Dependable Information Exchange for Next Generation Mobile Cyber-Physical Systems", IEEE Wireless Communications, vol. 24, no. 5, pp. 150-156, 2017.

[10] F. Ferrari, "Enabling dependable communication in cyber-physical systems with a wireless bus", TIK-Schriftenreihe, vol. 141, 2013.

[11] "Building Resource Adaptive Software Systems (BRASS) - DARPA-BAA-15-36 (Archived) - Federal Business Opportunities: Opportunities", Fbo.gov, 2017. [Online]. Available:https://www.fbo.gov/index?s=opportunity&mode=form&id=61f6223b1e9a85a0fc35d338d54621b6&tab=core&_cview=0. [Accessed: 19- Nov- 2017].

[12] "IT Complexity Report PDF - Trustmarque", Trustmarque, 2017. [Online]. Available: https://www.trustmarque.com/it-complexity-report-pdf/. [Accessed: 19- Nov- 2017].

[13] J. Garside, "Nasdaq crash triggers fear of data meltdown", the Guardian, 2017. [Online]. Available: http://www.theguardian.com/technology/2013/aug/23/nasdaq-crash-data. [Accessed: 19- Nov- 2017].

[14] T. Fuxreiter, C. Mayer, S. Hanke, M. Gira, M. Sili and J. Krofp, "A modular platform for event recognition in smart homes - IEEE Conference Publication", Ieeexplore.ieee.org, 2017. [Online].Available:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5556587&isnumber=5556525. [Accessed: 19- Nov- 2017].

[15] P. Dobrev, D. Famolari, C. Kurzke and B. Miller, "Device and service discovery in home networks with OSGi", IEEE Communications Magazine, vol. 40, no. 8, pp. 86-92, 2002.

[16] T. Hayet and K. Jilani, "A navigation model for a multi-robot system Based on Client/Server model," *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, St. Julian's, 2016, pp. 644-648.

[17] "Survey of prototyping solutions utilizing Raspberry Pi - IEEE Conference Publication", Ieeexplore.ieee.org, 2017. [Online]. Available:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7973568&isnumber=7973374. [Accessed: 19- Nov- 2017].

[18] L. Zhao, C. Ladas, and R. Edwards. A selective-ARQ scheme for improved TCP and UDP performance over wireless networks. AIMEE, 2017.

[19] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer Science & Business Media, 2009.

[20] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 2000.

[21] R. Nicholson, "Paremus Service Fabric – OSGi™ Alliance", Osgi.org, 2018. [Online]. Available: https://www.osgi.org/paremus-service-fabric/. [Accessed: 14- Jun- 2018].

[22] "Support Vector Machines", About Learning, 2018. [Online]. Available: https://amitranga.wordpress.com/machine-learning/support-vector-machines/. [Accessed: 14- Jun- 2018].