ISSN: 0976-3104
SPECIAL ISSUE: Emerging Technologies in Networking and Security (ETNS)
Narayanan and Arun

THE IIOAB JOURNAL

ARTICLE OPEN ACCESS

DISTRIBUTED POLICY TRACKING IN WIRELESS SENSOR NETWORKS USING PARALLEL EXECUTION OF TINYPOLICY OVER P2P NETWORKS' LIVE STREAMING VIDEO ON-DEMAND SERVICE

Madeshan Narayanan[1,2*] and Chokkalingam Arun[3]

[1]*Department of Computer Science and Engineering, Sathyabama University, Chennai, INDIA*
[2]*Department of CSE, Saveetha University, Chennai, Tamil Nadu, INDIA*
[3]*Dept of Electronics and Communication Engineering, RMK College of Engineering and Technology, Chennai, Tamil Nadu, INDIA*

ABSTRACT

Peer-to-Peer Networks (P2P) play a vital role in all major domains today. This paper discusses how P2P network services are extended to wireless sensor networks (WSNs). In a sensor node's local memory, Policy-based management applications can store only a restricted number of policies in the local memory of a sensor node on WSNs, subject to hardware resource constraints, and are required to be recycled whenever extra policies are needed. To handle these issues, an operation called Parallel Execution of TinyPolicy is developed for storing, locating, accessing and executing policies in a WSN. It is devised to make full use of the memory available in a P2P network and duplication. The result is a more robust policy system against any failure of nodes and single points. To govern and control the embedded devices, the parallel execution of TinyPolicy will facilitate WSNs to solve these difficult issues. Utilizing a P2P architecture, distributed policy-based management and replication of policies, the new framework offers several novel features like dynamic distribution of policies between the sensor nodes. Additionally, the parallel execution of TinyPolicy manages the location of these policies dynamically by thrusting the widely-used policies against the target node, instead of leaving them solitarily in the WSN. This framework is simulated by an NS-2 simulator. In the near future, a WSN-and-P2P system combination may be a guide to developing robust applications. A P2P platform provided by an abstract program for communicating and allowing developers to take charge of functionality is a sign of simplifying the development process of distributed applications.
.

Published on: 08th– August-2016

KEY WORDS

Distributed Systems; P2P Networks; Parallel Execution; TinyPolicy; Wireless Sensor Networks

***Corresponding author: Email:** narayanan_baba@yahoo.com; **Tel.:** +91 9381485813

INTRODUCTION

Wireless sensor networks (WSNs) are built for a variety of purposes. Their chief drawback, however, is that their parts have limitations in terms of processing and power. As a result, managing difficult algorithms with information collected by sensors ought to be made in modules outside the WSN. Application design could be a lot more flexible with the introduction of P2P networks. This can surmount the challenges usually related with WSNs as mentioned earlier. However, the design of distributed systems design is a challenging job, with a host of issues relating to communication between components.

In developing applications, P2P networks are a match for WSNs. While P2P networks are better suited for high-end nodes with substantial power, WSNs are well suited for capturing the surrounding information in intense conditions. The challenge is to integrate these architectures to cooperate for functionality's sake.

WSNs play an important role in many systems, assisting people in their usual day-to-day routines and getting them acclimatized to their present circumstances. But they are required to manage all by themselves to discover and configure tools for services, detect and respond to attacks, resolve faults and reconfigure the system to diminish all of these.

Based on the assessments of object attributes, a policy proclaims to choose target objects inside a domain. When all the objects in a domain can be applied by a policy, the simplest case arises. Domain membership can vary invariably and, consequently, a group of objects where the policy is applicable has to be examined during policy interpretation.

THE IIOAB JOURNAL
ISSN: 0976-3104

An illustration of a management authorization policy is the access rule, which indicates a connection between managers and managed objects in relation to managing operations allowed on objects of a certain kind. The rule also utilizes the scope to choose subsets of objects and, further, describe restrictions on these actions.

By stating why an object has been selected, a manager can indicate the primary membership for exploring a database of a different domain, except this is not offered as component of essential domain service. Policy membership is then required to restrict objects that can be consequently built into the domain or added from another. Additional membership policies are linked to the number of objects allowed in a domain.

A robust mobility-management structure for Internet Protocol version 6 (IPv6) and heterogeneous wireless networks, enabled by policy enforcement, is proposed. Policies are defined, based on infrastructure facility, service agreement and conciliation results. The results of the system's performance confirm that user experience has improved, largely in relation to connectivity. Service providers will therefore be able to hire expensive UMTS economically, and users will always be connected anytime, so long as the distribution of the flow and the selection of the network is visible and flexible [1,14].

This paper is organized in the following manner: In Section II, a step-by-step literature survey related to P2P network services on wireless sensor networks is tabled. In Section III, the proposed model for the parallel execution of TinyPolicy is presented. Details of the experimental results and implementation are presented in Section IV. The final section contains the conclusion and future extensions of the paper.

LITERATURE REVIEW

WSNs (Wireless sensor networks) have become all-encompassing in day to day life, penetrating into fields like environment, medicine and defence studies. Every WSN comprises of a number of sensors which are accountable for monitoring single or multiple events. A WSN generally operates in various environments where sensors are obtained from several manufacturers which lead to incompatible issues with respect to standards in hardware and software. Even though specific types of sensors may overcome a few of these issues, it comes at a cost with complexity issues. Researchers have, consequently, recommended policy-based management (PBM) platforms as a suitable solution to trounce these challenges and effectually camouflage the complicated workings behind basic network devices.

A starfish framework in sensor nodes targets self-healing policy deployment. The framework comprises a Finger2 policy system for dynamically adapting a library module to make programming the essential functions of nodes less complex, as well as a client-side editor to manage policies. The policies described are for a health-care body network, with self-healing features for sensor networks and re-configuring policies to handle faults. There is also an intention to broaden the concept of self-healing services and incorporate them in self-managed cell architecture [1]. A model based on a policy understanding the concerns of the several actors involved in practical WSN applications is recommended, achieved by enhancing and optimizing the runtime environment. A sample implementation of the model is implemented and examined, using the SunSPOT platform. The results show that the model is adequately lightweight and can be applied with great advantage in WSN environments. The focus will also be on interaction with several parallel programming models [2-3].

Managing several sensor nodes is a daunting task where energy concerns are an associated factor. Though numerous network structuring methods have been suggested, a system to cover the entire structure has yet to be recommended. A management setup for WSNs, known as the SNOWMAN framework, is built to address these shortcomings. It uses an approach based on a policy that lets sensor nodes organize and administer themselves independently. The effectiveness of this model is scrutinized using an NS-2 simulator. The results show that the suggested model permits lesser energy consumption and a longer lifetime than currently existing methods like the LEACH and LEACH-C [4].

Business-level operators can use methods like a policy-based network management (PBNM) to inscribe SLAs in a comprehensible interface without making changes to the codes executed in the controllers. A system for policy authoring to ease the process of configuring SDN architecture is set up. The framework aims at helping business-level operators easily indicate service needs, offer flexibility and permit the said operator to accept or reject suggestions. Even if the QoS classes increase, the toolkit functions well and within the intricacies of SLAs [5].

THE IIOAB JOURNAL
ISSN: 0976-3104

This paper compares, in different M2M environments, the performance of every PHY mode for IEEE 802.15.4g (SUN) and IEEE 802.11 (Wi-Fi P2P) WSNs. Performances, in terms of various configurations, were examined. The outcome shows that IEEE 802.11 is more susceptible than IEEE 802.15.4g in shadowing channels, relative to the AWGN channel. Hence, the FSK in IEEE 802.15.4g is promising. Whereas, in multipath fading channels, the performance of IEEE 802.11 was superior to that of IEEE 802.15.4g. On the basis of a suitable performance, $Eb/N0$, coverage of service and channel surroundings, an appropriate communication channel can be chosen [6].

CaPI (component and policy infrastructure), a dynamic component that can be reconfigured and acts as a piece of middleware for wireless sensor networks, is presented. CaPI offers two ideas for modern sensor network development and administration. One model encourages embedded developers, while the other supports managing and specifying behavioral apprehensions by managers or domain specialists. CaPI also allows successful customization and active re-configuration of the function and performance of applications. Runtime editions can be ratified effectively because of the survival of fine-grained and coarse-grained methods [7].

There are only a few protocols built for WSNs that are examined on real test beds. WSNs encounter challenges in the form of restricted power supply, little storage capacity, and connectivity problems. The virtual cord protocol (VCP) that utilizes P2P techniques is used for managing information in a WSN and offers effective resource usage. The execution and operation of the VCP as it reveals its performance in a real-time situation using Mica2 motes is discussed here, displaying how a P2P approach can be applied on WSNs for efficient data transport and management [8].

Based on the CIM policy model, a policy management for autonomic computing (PMAC) platform is developed. We provide here the PMAC platform's outline and how it can be utilized in managing network systems. The policy information model accepted by PMAC, and the model for communication between the resource and the policy manager, is presented. The key mechanisms of PMAC - for creating a policy, storing, evaluating and enforcing - are also presented, along with realistic applications of PMAC in managing networks [9].

Managing various nodes in huge numbers is always a tough task. A framework for a safe and policy-based administration of assorted resource-hampered networks of embedded systems is considered. Priority is given to the security of corresponding requests and responses. Messages are transmitted on a web-based approach and, additionally, provided a range of options for secure transmission. The methods used for this purpose depend on the application's requirements [10].

The design and execution of a management system known as SRM (sensor reliability management) for managing the reliability of data in WSNs are explained here. Though designed largely for managing the reliability of the data, it can be simply incorporated in various management services. SRM is made up of modules like user policy specification, evaluation, decision-making and action.

SRM further permits network administrators to communicate with the network by offering management policies. Results prove that SRM not only satisfies reliability requirements, but also decreases energy consumption by half [11].

It is hard to identify, access and administer a sensor node since WSNs normally work in dissimilar environments. There is, consequently, a need to surmount these challenges. The purpose here is to develop a new framework for managing policies in WSNs in a distributed fashion. The proposed work focuses on extending the functionality of WSN management by increasing policy numbers in WSN storage. It also masks the intricacies of policy management processes from users by streamlining those procedures [12]. Wireless sensor networks (WSNs) have inadequate hardware resources, thereby restricting management capabilities and causing volatility and irregularity in the system. The purpose of this work is to build a novel framework for policy-based management for wireless sensor networks (WSNs) to surmount the shortcomings of the current policy-based WSN platforms. The framework comprises major software elements like the monitor, LPDP (local policy decision point), PEP (policy enforcement point), and a set of integrated applications. The framework also consists of a 6-data warehouse [13].

WSN components have restrictions relating to power and processing. As a result, running complex algorithms has to be done using external components. Therefore, we present an amalgamation of WSNs and P2P networks to build systems relying on WSN functions. A programming concept is proposed that permits developers to focus on the operation of the developing method. Using feedback loops as a design tool, and the development of the concept's components, are also suggested. Further, they are required to be compatible, extensible and lowly-fixed [15].

THE IIOAB JOURNAL
ISSN: 0976-3104

One of the key components of WSNs is PBMS (policy-based management systems). Owing to hardware resource limitations, only a partial number of policies can be stored in a sensor node's local memory by policy-based management applications on WSNs, needing to be recycled if extra policies are needed. To handle this particular issue, a framework known as TinyPolicy is built using a P2P policy storage and operating system called PolicyP2P. It is developed to make use of the memory available on the network as a result of which the policy mechanism is tougher against failure of nodes and solo failure points [16].

PROPOSED MODEL ON PARALLEL EXECUTION OF TINYPOLICY

The novel policy creation method is started via a policy-user interface on a system connected to the WSN's source peer, which is a highest-level node in the P2P hierarchy address structure. Normally, this peer is the WSN's gateway node and possesses ample power and memory.

Figure-1 represents the Proposed Model for Parallel Execution of TinyPolicy Diagram and the directions for making such a policy are also described. Once the policy is generated by utilizing the Policy GUI, the source peer employs the P2P software part to establish a host node address for the policy. It executes this by building a policy key (policy ID = event ID + sequence ID + session ID) and then messing the policy key to fit inside the sensor network's address space. The source peer's monitor will push the hashed policy key against the node with the best matching address.

After the triggering process, policies are executed in a parallel mode, having already been simplified into minuscules. Policy execution starts once the policy request is made from the required node. The request generally happens from the root node or a remote adjacent node. If it does not exist, the request is then forwarded to the next node and likewise to the root node. Two types of policy repositories are available in this system: a dependent policy repository and an independent policy repository. A dependent policy repository contains dependent policies. For example, a policy 'A' writes two lines in a particular node and policy 'B' writes an additional two lines in that node (with the first two lines of 'A' and the next two new lines of 'B,' we consider that A and B are dependable). If the requested policy is dependable, it searches the dependable repository and if it is undependable, it searches the undependable repository. The requesting time and search time, consequently, are minimized. The question then arises as to how to ascertain whether or not the requested policy is dependable or undependable.

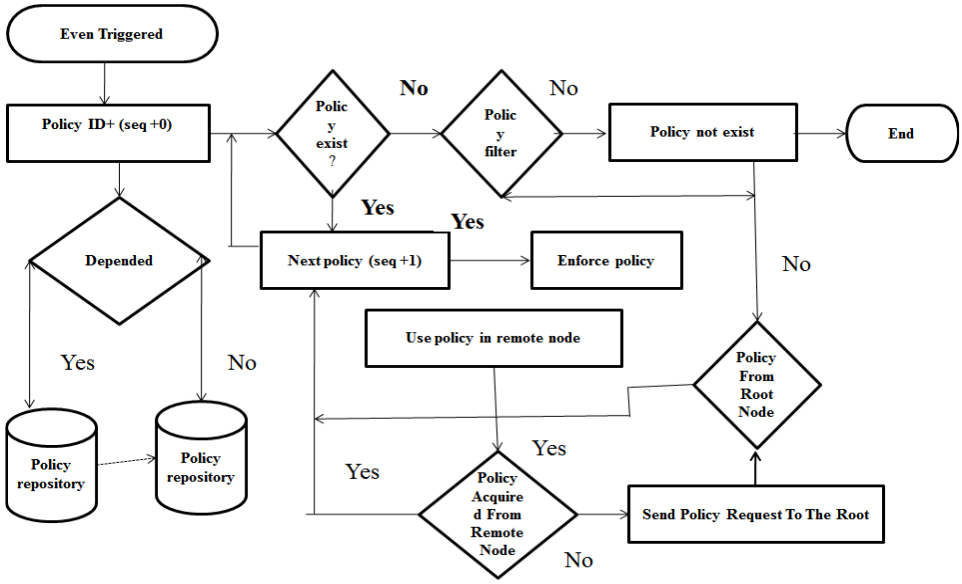

Fig:1. Proposed Model for Parallel Execution of TinyPolicy Diagram

Policy Creation

Policy ID = Event ID + Sequence ID + Session ID

THE IIOAB JOURNAL
ISSN: 0976-3104

= (28+1) x (28+1) + 28+1 + 2 byte
= 66049 + 257 + 2 byte
= 2 byte + 1 byte + 2 byte = 5 byte

Event Id – sub task
Sequence Id – 10 kb ----each 10kb send to
Session id: up completing one work

RESULT AND IMPLEMENTATION

Tiny Policy Implementation

We consider 25 nodes in NS-2 simulation. Nodes that try actions such as receiving, sending or storing data need policies. Policies have already been stored in two types of storage areas, dependable and undependable data sources. If a node needs a policy to perform an action, it requests a neighbor node. This request is forwarded to the root node via intermediate nodes. The policy filter module also works to detect whether or not the policy already exists in the intermediate node. If it exists, then it simply forwards the request to the policy hold node; otherwise, it forwards it to the root node. The policy filter also checks whether the required policy is in dependable storage or in undependable storage. The policy is now executed parallelly, with the time taken for overall execution of the policy reduced. The result obtained in this simulation is tabulated below.

Policy Replacement

Consider a case where the root node sends a video file to the request node at time t1. At that point in time, the root node creates a policy with a session id and destination id. A node requesting the same video file after a while is considered to be at time t2. Now, since the policy is the same, simply update it with the name of the session id and destination id, the policy being stored in two repositories, dependable and undependable. Dependable means that the policy regards current action while undependable means that it does not. Old policies, consequently, are stored in undependable repositories from which a policy can be taken and simply updated with the session id and destination id. This architecture helps in reusing, with certain modifications, policies.

According to the policy structure, we can easily update a current policy from an old one.
To update an old policy, we use a pattern-matching algorithm. The **[Table- 1]** represents parameter and size, the **[Table- 2]** shows policy table with action and data size.

Table: 1.Parameter and Size

Policy Structure	Data size
[ID] Policy id	5 byte
[IF] Policy condition	3 byte
[THEN] |Do Policy Action	3 byte
[END] End Policy Execution	1 byte
[NEXT] Execute Next Policy	3 byte

Frequent and Closed Patterns

Table: 2. Policy table with action and data size

Policy	Action	Data size
P1	A1	D1
P2	A1	D2
P3	A1	D1
P4	A2	D2
Pn	An	Dn

THE IIOAB JOURNAL
ISSN: 0976-3104

Where A1, A2, A3 …An is action, D1, D2, D3….Dn
Is data size, and P1, P2, P3….Pn is policy.
Both P1 and p3 are equal. Notwithstanding the fact that they are equal, they do not create p3, since P3 is an existing policy

Frequent pattern covering set

Table: 3. Mapping, destination and frequent action

Frequent Pattern	Destination
{ A1, A2,A3}	{ d1, d2, d3…..dn }
{ A2}	{ d2, d3…..dn }
{ A2,A3}	{ d1, d3…..dn }

Wherein A1, A2, A3…An is action and d1, d2, d3….dn
Is destination. The **[Table- 3]** shows mapping of destination and frequent pattern.

Determining the Requested Policy Existence

FREQUENT and CLOSED PATTERNS
TERMSET(Y) = {t/¥ dn ζ Y □ t ζ dn}
CLOSURE of X is defined
CLOSURE(X) = TERMSET(X1)
A PATTERN X; if, and only if, X = CLOSURE(X)
Let X be a CLOSED PATTERN.
We can prove that
supa(X1)<supa(X)
FOR all PATTERNS X1 ⊐ X; OTHERWISE, if supa(X1)=supa(X)
We have X11=X1
Wherein supa(X1) and supa(X) are the total support of PATTERN X1 and X.
Also we have
CLOSURE(X) = TERMSET(X1) = TERMSET(X11) ⊐ X1 ⊐ X
That is, Closure(X) ≠ X.

Evaluation of policy in terms of number and time

Table: 4. Evaluation of the Policy in Terms of Number and Time

Number of policy	Evaluation time (traditional method)	Evaluation time (proposed method)
0	0.6	0.2
1	1.2	1.1
2	1.4	1.6
3	2.2	1.8
4	3.2	2.0
5	3.8	2.6
6	4.2	3.1
7	4.8	3.5
8	5.2	4.1
9	5.7	4.6
10	6.2	5.1

THE IIOAB JOURNAL
ISSN: 0976-3104

[Table- 4] represents the evaluation of the policy in terms of number and time. This graph mentions the time of execution in traditional methods and the proposed method between TinyPolicies. Figure-2 shows 5 traditional policies completed within 3.8 seconds but 5 proposed policies completed within 2.6 seconds; and 10 traditional policies completed within 6.2 seconds, with 10 proposed tiny policies completed within 5.1 seconds. An overall indication from the graph is that parallel execution increases the number of policies within a given time, also helping increase the overall performance of the system.

Evaluation of the policy in terms of size and implementation time

[Table- 5] represents the evaluation of the policy in terms of its size and implementation time. Figure-3 mentions the time of execution between TinyPolicies in traditional methods and the proposed method. It shows that 200-byte sized traditional policies completed within 0.2 seconds, but 300 bytes of policies completed within 2.6 seconds, in both cases. 1000 traditional policies completed within 2.0 seconds and, similarly, 10 proposed TinyPolicies completed within 5.1 seconds. In 5.2 seconds, we discover that both cases have nearly 1000 different policies executed between them. An overall indication from the graph is that parallel execution increases the number of policies within a given time, also helping increase the overall performance of the system.

Table: 5.Evaluation of the Policy In Terms Of Size and Execution Time

Policy size (byte)	Evaluation time (traditional method)	Evaluation time (proposed method)
200	0.6	0.2
300	1.2	1.1
500	1.4	1.6
700	2.2	1.8
1000	3.2	2.0
1200	3.8	2.6
3000	4.2	3.1
3200	4.8	3.5
3600	5.2	4.1
4000	5.7	4.6
5000	6.2	5.1

CONCLUSION

An extension of the services of P2P networks for wireless sensor networks (WSNs) is evaluated in this paper. The downside of WSNs is that their components are restricted in terms of power and processing, due to which policy-based WSN administration applications can only store a particular amount of policies in the limited memory of sensor nodes. Challenging concerns in relation to govern and control the embedded devices can be resolved by WSNs with the help of the parallel execution of TinyPolicy. P2P networks are fairly flexible in designing applications and can address most of the problematic issues that confront WSNs. Integration could be the key challenge, between these architectural structures, for cooperating in a specific functionality. The new structure uses P2P architecture, distributed policy-based management and a replica of policies for dynamically distributing policies between sensor nodes, a feature supported by this framework. The location of policies can also be managed dynamically. The framework is simulated using the NS-2 simulator. For simulation, 25 nodes are considered. A node planning to perform actions like receiving, sending or storing data requires policies, usually stored in two types of storage: dependable and undependable. Nodes request a neighbor node for a policy to execute an action. This request is then forwarded to the root node through intermediate nodes. The policy filter module detects whether or not the policy already exists in the intermediate node. It also examines if the policy is in dependable or undependable storage. In future, a combination of WSNs and P2P computing may result in the development of robust applications. From a future perspective, we intend to move towards parallel steps to find the existing policy and a new design, with whichever being completed first taking effect. The algorithm will be designed with the same perspective as well.

THE IIOAB JOURNAL
ISSN: 0976-3104

ACKNOWLEDGEMENT

We would like to express my deepest gratitude to Saveetha School of Engineering; the Management of Saveetha University; the Principal and Head of the Department of Computer Science & Engineering.

CONFLICT OF INTEREST

No conflict of interest

FINANCIAL DISCLOSURE

No financial support was received to carry out this project.

REFERENCES

[1] Shen, Chong, Wencai Du, Robert Atkinson, and Kae Hsiang Kwong. [2012]Policy based mobility & flow management for IPv6 heterogeneous wireless networks. *Wireless Personal Communications* 62(2): 329-361.

[2] Matthys, Nelson, Christophe Huygens, Danny Hughes, Jó Ueyama, Sam Michiels, and Wouter Joosen.[2011] Policy-driven tailoring of sensor networks. *In Sensor Systems and Software*, pp. 20-35. Springer Berlin Heidelberg,

[3] Bourdenas, Themistoklis, and Morris Sloman.[2010] Starfish: policy driven self-management in wireless sensor networks. In Proceedings of the 2010 ICSE *Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 75-83. ACM,.

[4] Si-Ho, CHA, LEE Jong-Eon, JO Minho, Hee Yong Youn, KANG Seokjoong, and CHO Kuk-Hyun. [2007]Policy-based management for self-managing wireless sensor networks. *IEICE transactions on communications* 90(11) :3024-3033.

[5] Machado, Cristian Cleder, Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, and Alberto Schaeffer-Filho.[2015] Policy authoring for software-defined networking management. In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, *IEEE* pp. 216-224.

[6] Oh, Eui-Suk, Seong-Hee Lee, Seong-Hyeong Lee, and Seung-Hoon Hwang.[2015] Comparison of SUN and Wi-Fi P2P WSN in M2M Environments. *International Journal of Distributed Sensor Networks* 501 (2015): 791849.

[7] Matthys, Nelson, Christophe Huygens, Danny Hughes, Sam Michiels, and Wouter Joosen.A component and policy-based approach for efficient sensor network reconfiguration. In Policies for Distributed Systems and Networks (POLICY), 2012 IEEE International Symposium on, pp. 53-60. *IEEE*, 2012.

[8] Sahar, Syeda Nida, Faisal Karim Shaikh, and Sana Hoor Jokhio.[2011] P2P based Data Management WSNs: Experiences and Lessons Learnt from a Real-world Deployment. *Mehran University Research Journal of Engineering & Technology* 30(4)

[9] Agrawal, Dakshi, Kang-Won Lee, and Jorge Lobo. [2005]Policy-based management of networked computing systems. Communications Magazine, *IEEE* 43(10) :69-75.

[10] Rantos, Konstantinos, Alexandros Papanikolaou, Konstantinos Fysarakis, and Charalampos Manifavas.[2012] Secure policy-based management solutions in heterogeneous embedded systems networks. In Telecommunications and Multimedia (TEMU), 2012 International Conference on, pp. 227-232. *IEEE*.

[11] Le, Tuan D, Wen Hu, Sanjay Jha, and Peter Corke.[2008] Design and implementation of a policy-based management system for data reliability in wireless sensor networks. In Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on, pp. 762-769. *IEEE*.

[12] Qwasmi, Nidal, and Ramiro Liscano. [2012]Framework for Distributed Policy-Based Management in Wireless Sensor Networks to Support Autonomic Behavior. *Procedia Computer Science* 10 : 232-239.

[13] Qwasmi, Nidal, and Ramiro Liscano. "Distributed Policy-Based Management for Wireless Sensor Networks. *Procedia Computer Science* 10 (2012): 1208-1212.

[14] C. Rajan, N. Shanthi, "Swarm optimized multicasting for wireless network", Life Sci. J, Vol.4, No. 10, 2013.

[15] Gutierrez, German, Boris Mejías, Peter Van Roy, Diana Velasco, and Juan Torres. [2008] WSN and P2P: a self-managing marriage.In Self-Adaptive and Self-Organizing Systems Workshops, 2008. SASOW 2008. Second IEEE International Conference on, 198-201. *IEEE*

[16] Qwasmi, Nidal, and Ramiro Liscano.[2015]] TinyPolicy: A distributed policy based management framework for Wireless Sensor Networks. In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, pp. 918-921. *IEEE*,

ABOUT AUTHORS

M. NARAYANAN *received a B.E. in Computer Science & Engineering from Anna University, Chennai, in 2006, and an M.E. (with distinction) in Computer Science & Engineering from Sathyabama University, Chennai, in April 2010. He has also been pursuing a Ph.D. in Computer Science & Engineering at Sathyabama University since July 2010. Presently he is Assistant Professor, Saveetha School of Engineering, Saveetha University, Chennai, and Tamil Nadu. He has 9 years' teaching experience, with research interests in distributed systems, peer-to-peer networks, and video-on-demand service.*

Dr. C. ARUN *received a B.E in Electronics & Communication Engineering from Anna University, Chennai, as well as an M.E and Ph.D, in Image Processing from Anna University, Chennai, Tamil Nadu, and India. He is, currently, Professor in the Department of Electronics & Communication Engineering, RMK College of Engineering & Technology, Chennai, Tamil Nadu, and India. With 15 years' experience in teaching engineering in several colleges, his research interests lie in image processing, cloud computing, distributed systems, and peer -to-peer networks.*