International Journal of Computer Networks (IJCN)

ISSN : 1985 - 4129

Volume 2 - Issue 6
Number of issues per year: 6

International Journal of Computer Networks (IJCN)

Volume 2, Issue 6, 2011

Edited By
Computer Science Journals
www.cscjournals.org

Editor in Chief Associate Professor Min Song

International Journal of Computer Network (IJCN)

Book: 2011 Volume 2, Issue 6
Publishing Date: 08-02-2011
Proceedings
ISSN (Online): 1985-4129

IJCN Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers

Editorial Preface

The International Journal of Computer Networks (IJCN) is an effective medium to interchange high quality theoretical and applied research in the field of computer networks from theoretical research to application development. This is the third issue of volume second of IJCN. The Journal is published bi-monthly, with papers being peer reviewed to high international standards. IJCN emphasizes on efficient and effective image technologies, and provides a central for a deeper understanding in the discipline by encouraging the quantitative comparison and performance evaluation of the emerging components of computer networks. Some of the important topics are ad-hoc wireless networks, congestion and flow control, cooperative networks, delay tolerant networks, mobile satellite networks, multicast and broadcast networks, multimedia networks, network architectures and protocols etc.

IJCN give an opportunity to scientists, researchers, engineers and vendors to share the ideas, identify problems, investigate relevant issues, share common interests, explore new approaches, and initiate possible collaborative research and system development. This journal is helpful for the researchers and R&D engineers, scientists all those persons who are involve in computer networks in any shape.

Highly professional scholars give their efforts, valuable time, expertise and motivation to IJCN as Editorial board members. All submissions are evaluated by the International Editorial Board. The International Editorial Board ensures that significant developments in computer networks from around the world are reflected in the IJCN publications.

IJCN editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCN. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJCN provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Computer Networks (IJCN)

Editorial Board

Editor-in-Chief (EiC)

Associate Editors (AEiCs)

Editorial Board Members (EBMs)

Table of Content

Volume 2, Issue 6, July 2011.

Pages

Energy Behavior in Ad Hoc Network Minimizing the Number of Hops and Maintaining Connectivity of Mobile Terminals Which Move from One to the Others

Kohei Arai arai@is.saga-u.ac.jp
Department of Information Science, Faculty of Science and Engineering
Saga University
Saga, 8400021, Japan

Lipur Sugiyanta lipurs@gmail.com
Department of Electrical Engineering,
Faculty of Engineering
State University of Jakarta
Jakarta, 13220, Indonesia

Abstract

Wireless ad-hoc network is a special kind of network, where all of the nodes move in time. The topology of the network changes as the nodes are in the proximity of each other. Ad-hoc networks are generally self-configuring no stable infrastructure takes a place. In this network, each node should help relaying packets of neighboring nodes using multi-hop routing mechanism. This mechanism is needed to reach far destination nodes to solve problem of dead communication. This multiple traffic hops within a wireless ad-hoc network caused dilemma. Wireless ad-hoc network that contain multiple hops become increasingly vulnerable to problems such as energy degradation and rapid increasing of overhead packets. This paper provides a generic routing framework that balances energy efficient broadcast schemes in Wireless Ad-Hoc Network and maintaining connectivity of nodes (mobile terminals). Typically, each node's activities will consume energy, either for sending packets, receiving or preparing/processing packets. Number of hops, distance of nodes, and size of packet will determine the consumption of energy. The framework is based on the principle that additional relay nodes with appropriate energy and routing metric between source and final destination significantly reduces the energy consumption necessary to deliver packets in Wireless Ad-Hoc Network while keep the connectivity of dynamic nodes. Using the framework, the average network connectivity is kept 18% higher and the lifetime of network lasting more than 2.38% compared with network with Link State Routing mechanism. The simulation notes that the end-to-end delay may increase rapidly if relay nodes are more than five.

Keywords: Multi-Hops, Energy, Connectivity, Metric

1. INTRODUCTION

A Wireless Ad-Hoc Network consists of mobile nodes platforms which are free to move in the area. Node is referred to a mobile device which equipped with built-in wireless communications devices attached and has capability similar to autonomous router. The nodes can be located in or on airplanes, ships, cars, or on people as part of personal handheld devices, and there may be multiple hosts among them. Each node is autonomous. The system may operate in isolation, or have gateways to a fixed network. In the future operational mode, multiple coverage of the network is expected to operate as global “mobile network” connecting to legacy “fixed network”.

At each time and every node’s positions, a wireless connectivity in the form of a random, single-hop, and multi-hop paths may exists among nodes. This topology may change as the nodes move or adjust their parameters. Among networks, Wireless Ad-Hoc Network has several characteristics, e.g.:
1) Dynamic topologies,
2) Bandwidth-constrained,
3) Energy-constrained operation, and
4) Limited physical security.

These characteristics create a set of underlying assumptions and performance considerations for protocol design which extend beyond static topology of the fixed Internet. All nodes in Wireless Ad-Hoc Network rely on batteries or other exhaustible energy modules for their energy. For this network, the most important system design criterion for optimization is energy conservation. Thus one critical design issue for future Wireless Ad-Hoc Network is the development of efficiently power consumption that suits communication architectures, protocols and services of network enabled wireless devices. Energy conservation means to maximize the operational lifetime of a node, thus, enhancing the overall user experience [12].

In this paper, we provide a framework that balances between energy efficient broadcast schemes in Wireless Ad-Hoc Network and maintaining connectivity. Various formation options of nodes and their potential overheads and impacts on efficiency are evaluated via simulation study. A comparison between similar network of Link State Routing and the generic framework is also conducted. Simulation results show that modified algorithms under different formation conditions are more efficient than the previous one. The remainder of this paper is organized as follows: Section 2 gives preliminaries and our system model. Section 3 discusses the detail design of the simulation model, its notations, and assumptions. Simulation algorithm that suits mobile environment is presented in Section 4. A performance evaluation of generic algorithms and comparison to a broadcast-based link state routing protocol that uses transmission power as the link cost unit are presented in Section 5 and Section 6, respectively. Section 7 concludes the paper.

2. PRELIMINARIES

Wireless network configuration is generally set up with a centralized access point for provide high level of connectivity in certain area. The access point has knowledge of all devices in its area. Routing to nodes is designed in a table driven manner [14][17][18]. The [17] introduced a technical review of wireless network technology products that implemented IEEE802.11 standard through experiments of fixed wireless network nodes. In terms of review the network performance at this stage, it will be represented as the view of use and evaluation of outdoors Muni-WiFi devices in accordance to applying the legacy LAN technology inside the corporate network. Performance of network access layer, e.g. performance of voice and TCP data transmission in terms of throughput, response time between nodes, and communication delay in multi-hop transmission are presented.

However, the [17] were intended to operate in static topology network. With recent performance in computer and wireless communications technologies, advanced mobile wireless is expected to see increasingly widespread use and application. The vision of future mobile ad hoc networking is to support robust and efficient operation in mobile wireless networks by incorporating routing functionality such that networks are capable to be dynamic, rapidly-changing with random, multi-hop topologies which are likely composed of relatively bandwidth-constrained wireless links. Supporting this form of host mobility requires address management, protocol interoperability enhancements and the like.

In this dynamic network, broadcasting plays a critical role especially in vehicular communication where a large number of nodes are moving and at the same time sending a large size of packet. In wireless network where nodes communicate with each other using broadcast messages, the broadcast environment works as receivers collect information from all transmitting nodes within its coverage pattern's neighborhood, and then allowing receivers to aware of immediate surrounding respond before re-transmitting packet [9][10][4]. Several transmissions may be unnecessary during broadcasting mechanism. These redundant cause the broadcast storm problem [22], in which redundant packets cause contention and collision consume a significant percentage of the available energy resources. Thus, routing protocols should be capable to respond these changes using minimum signaling and taking into account the energy as a parameter distributed in network.

To address these challenges, we propose framework of energy aware broadcasting technique for wireless ad hoc networks. The framework uses a packet forwarding technique where neighbor nodes can be elected to be relay on behalf of source-destination path with the goal of optimized the overall energy consumption to deliver packets in the network, while maintaining the connectivity among nodes. Transmission to a distant node may consume a higher amount of energy in comparison to transmission to a node in closer range and more energy will be used for sending packets than receiving or processing packets. In addition, transmission of larger packets may consume a higher amount of energy in comparison to transmission of smaller packets. The framework is based on the principle that adding additional relay nodes with appropriate energy and routing metric between source and final destination nodes significantly reduces the energy consumption necessary to deliver packets in Wireless Ad-Hoc Network while maintaining connectivity among nodes.

3. SIMULATION MODEL, NOTATIONS, AND ASSUMPTION

Relays are intermediate nodes between source and final destination which help relaying packets using multi hop connectivity mechanism. The appearance of relays is required to avoid dead communication if the distance is not in the proximity of source node. Multi hop connectivity can extend the mobility and expand the coverage area, but in the same time increase the delay time. In general, the more relay nodes (hop), the longer the delay time. Energy consumption is also affected. Direct transmission is seemed to have more aggregate energy required than indirect communication. Thus the adjustment of relay nodes will influence the balance of delay time and energy consumption.

The simulation is initiated from broadcast mechanism and propagated through node-to-node based routing approach. During propagation, it takes into account both data transmission and route discovery. This model is not only applicable in direct communication (one hop transmission) but it can also work in multi-hop transmission. In this situation, when the source and final destination nodes are located outside the maximum transmission range, source node is capable to discover multiple hop routing efficiently thus maintain the energy level required in comparison to standard flooding based ad hoc routing designs.

The Model

Simulation describes that antenna module installed in each node is capable of dynamically adjusting the transmission energy used to communicate with other nodes. Industrial standard of antenna module that support IEEE 802.11 include a management for controlling this energy consumption. Simulation assumes that the energy consumption required to transmit a packet between nodes A and B is similar to that energy required between nodes B and A if and only if the distance and the size of packet are same. The coverage distance range of the nodes is a perfect symmetric unit disk (omnidirectional). If $d_{x,y} \leq r_x \rightarrow$ x and y can see each other. This assumption may be acceptable in the condition that interference in both directions is similar in space and time; which is not always the case [7]. Usually interference-free Media Access Control (MAC) protocol such as Channel Sense Multiple Access (CSMA) may exist. In addition, wireless link channel is assumed to have no physical noise; i.e., the errors in packet reception due to fading and other external interferences are not considered as a serious problem. Packets from sender to receiver will be transmitted as long as the bandwidth capacity is sufficient and the received signal to noise ratio (SNR) is above a certain minimum value. Thus every packet successfully received is acknowledged at the link layer and de-encapsulate at the higher layer. Each node is capable of measuring the received SNR by analyzing overheard packet. A constant bit error rate (BER) is defined for the whole network. Whenever a packet is going to be sent, a packet's CRC is generated. At the receiver, packet's CRC will be checked; if the random number generated is greater than the CRC value of received packet, then message is received, otherwise it is lost. The default value for the BER is 0, which means there is no packet loss due to physical link error.

Simulation cover a single area of homogeneous nodes that communicate with each other using the broadcast services of IEEE 802.11. There are nodes with different roles simulated in this simulation, namely initiator node/source node, receiver node, sender node, destination node, and final destination node. Initiator node/source node is node that initiates transmission of packets. Packet can be either route discovery or data transmission. Like other nodes, initiator is always moving with random direction, speed, and distance. At the time it is moving, initiator node is always sensing its neighbor to maintain connectivity. Receiver node is node that can be reached by source/sender node. Nodes are defined as neighbors if it located within its distance radius range. At initial time, node senses its neighbors before packet data is required to be transmitted. Coverage neighbor nodes always receive packets that are broadcasted from sender. Destination node is selected receiver node in multi hop transmission that should relay packets to the next receiver node. Final destination node is node that became the end destination of packets.

The layered concept of networking was developed to accommodate changes in local layer protocol mechanism. Each layer is responsible for different functions of the network. It will pass information up and down to the next subsequent layer as data is processed. Among the seven layers in the OSI reference model, the link layer, network layer, and transport layer are 3 main layers of network. The framework is configured mainly in those layers. Genuine packets are initiated at Protocol layer, and then delivered sequentially to next layer with fragmentation process of selected packets. These fragmented packets are to be randomly distributed. Simulation models at every layer owned with finite buffers. Limited buffer makes packets are queued up according to the drop tail queuing principle. When a node has packets to transmit, they are queued up provide the queue contains less than K elements ($K \geq 1$). To increase the randomization of the simulation process, simulation introduces some delay on some common processes in the network, like message transmission delay, processing delay, time out, etc. This behavior will result that at each instance of a simulation would produce different results. The packets exchanged between sender and receiver is designed at a fixed rate transmission λ based on a Poisson distribution. Nodes that have packet queued are able to transmit it out using in each available bi-directional link channels.

Energy is power kept in each node. [7] was assumed that the radio dissipates E_{elec} = 50 nJ/bit to run the transmitter or receiver circuitry and ε_{amp} = 100 pJ/bit/m^2 for the transmit amplifier. The radio model is shown in the Figure 1 below.

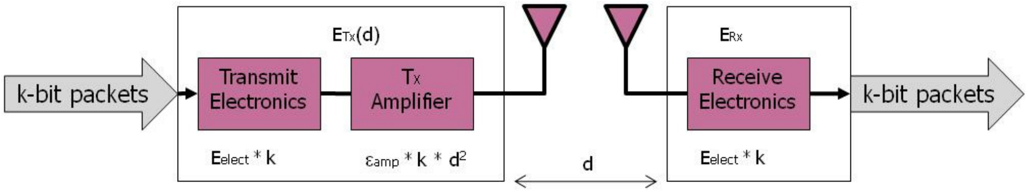


FIGURE 1: The radio model.

Thus, to transmit a k-bit message a distance d using this radio model, the radio expends:

$$E_{Tx}(k,d) = E_{Tx-elect}(k) + E_{Tx-amp}(k,d) \tag{1}$$

$$E_{Tx}(k,d) = E_{elect} * k + \epsilon_{amp} * k * d^2$$

and to receive this message, the radio expends:

$$E_{Rx}(k) = E_{Rx-elect}(k) \tag{2}$$

$$E_{Rx}(k) = E_{elect} * k$$

The energy behaviors of node are defined as follow:

- During the idle time, a node does not spend energy. Even though this assumption has been proven untrue because being idle might be as costly as receiving data, this is still an assumption that can be done in most experiments, since the most important factor is the overhead in terms of message exchange and its associated cost.
- The nodes are assumed to have one radio for general communication. The main radio is used in all operations when the node is in active mode, and to send and receive control packets. When this radio is turned off, then no messages will be received and no energy will be used.
- Energy distribution among nodes can either be constant value, normally distributed, Poissonly distributed, or uniformly distributed.

Algorithm

The primary goal of this research is to balance between the energy level required during data transmission in the network and maintaining connectivity to the others. A node keeps its transmitter “on” to transmit one data packet to another node for L/V seconds where L is the size of the transmitted packet in bits (i.e., data plus headers) and V is the raw speed of the wireless node in m/second. Similarly, the receiver node keeps its transmitter “on” to acknowledge a successful data transmission for an associated period of L/V seconds.

As foundation for this mobile environment, the core algorithm is developed from static mode (e.g., sensor networks). The enhancement algorithm for serving mobility then detailed in support of topology development, topology maintenance, and routing maintenance.

Topology development involves transmit and receives of HELLO packets, REPLY packets, and so on; mostly redundant. These packets that successfully received by link layer will update an entry in the table (neighbor table) which cache information about surrounding nodes exist. HELLO packets and corresponding REPLYs have contents of [ID, hop, energy, time, initialTime], where ID is a unique neighbor node (IP address), hop is a number which increment each time packet reach at relay node, energy is current available energy level needed to ensure the communication with the neighbor node, time is current time at which this event is executed, and initialTime is time from which this event was generated.

Let $\mathbf{E}_{min\_i}$ is the minimum energy ratio of node **I** at which a node can still receive, process, and transmit packets. Node **j** finds out the energy level of neighbor node **I** through analyzing of received reply packet from node **I** as it responded the previous transmitted Hello packets. Thus the computation of $\mathbf{E}_{min\_i}$ is done through two-step propagations. The use of two-steps propagation model is to simulate interactive propagation in the operation of the protocol in dynamic environment. As a future research, the appropriate propagation model that best matches to this environment should replace the simple two-steps model presented here [9][10][4]. The two-steps propagation model is appropriate for outdoor environments where a line of sight communication existed between the transmitter and receiver nodes and when the antennas are omni-directional. The two-steps propagation model assumes there are two main signal components. The first component is the signal traveling on the line of sight to reached neighbors along with its reply from neighbors and the second component is a confirmation packet transmitted to selected neighbors.

The topology maintenance algorithm is responsible for performing the route optimization operation that leads to respond of routes changes. The route maintenance algorithm performs two basic operations: initiate broadcast maintenance packets, which computes either a route optimization between two nodes is needed and sets up broadcast mechanism; and executes maintenance packets, which determines when to transmit routing maintenance packets.

Because several relay nodes may exist between source and final destination node, multiple packets are sent to only a (selected) single next relay node. The source or sender node will choose the one providing a higher metric value (a combination of energy metric and distance metric). Only one certain node between two communicating nodes can be selected as a relay node at a time. From the simulation, it noted that transmission of multiple route-redirect packets wastes bandwidth and power resources. For sparsely populated networks, this may not be a problem. However, this is an issue in the case of densely populated networks where several potential nodes can be chosen. The framework addresses this issue by giving priority for the execution of an update routing maintenance packet to the potential neighbor node that owned highest route metric energy-distance values first. After receiving an update topology maintenance packet, a node modifies its routing table, putting the source of the received packet as the next hop node for the specific sender-destination route path.

To execute preferential event in sequentially distributed events, we used a simple approach that consists of applying a different time-event execution by means of the triggering event sequences action. The lower and upper bound of the queuing interval are set such that they do not interfere with predefined timers used by the other events for layers and modification events.

Network Convergence

The steps to optimize network topology are iteration of the same procedure of determining relay nodes between source and final destination. At first time, source node transmits packets with TCP type in bursts to neighbor nodes to start the sequence events of topology development. The further the distance the more relay nodes are added.

The framework optimizes routes through sequence of steps to converge to an optimum route. The step refers to the event in which the simulation initiates a source node to transmit a Hello request for the first time. The network will converge as fast as the transmission speed of packets transmitted by node. The more explicit process is explained below.

At the first iteration, the source node communicates directly with the destination node. All destinations in proximity will receive the packets due to the broadcast nature of wireless communication media. Upon receiving a packet, destinations create reply packet which contain its condition (i.e. energy and its position in neighborhood map) and sent it back to sender. Selected destinations also forward the packet to the next neighbors. During this sequence, relay node is determined and confirmed by relevant information gathered at each group of neighborhood nodes. After omitted redundant packets and based on calculation metric value, relay node is set (i.e., a small set of nodes that potentially forward the broadcast packet) to achieve high delivery ratio with energy consideration. The new relays added to a route during iteration are very much dependent on the relay nodes found in the previous iteration. The set can be selected dynamically (based on both the connectivity and broadcast state information). This relay node set forms a connected dominating set (CDS) and achieves full coverage of connected network. It is possible that the first iteration, which seemed as most optimum value of metric value is not the route achieving the optimum energy-routing path.

We built network simulator to evaluate this performance. The simulator supports physical, link and routing layers for single/multi hop ad-hoc networks. We assume that IEEE 802.11 Distributed Coordination Function (DCF) or MAC protocol which uses Channel Sense Multiple Access with Collision Avoidance (CSMA/CA) already deployed. Successfully received packet by receiver's interface is packet whose SNR is above a certain minimum value otherwise the packets cannot be distinguished from background noise/interference. Packets are transmitting through physical layer in accordance with Poisson distribution. Communication between two nodes in IEEE 802.11 uses TCP signaling before the actual data transmission takes place. Simulation simulates this with random hearing to link's condition. If link allow packets to be sent, then sender executes some packets already queued.

4. CONSIDERATION OPERATION IN MOBILE ENVIRONMENT

Update energy metric and distance metric values in the mobile environment uses a propagation model. The propagation model is based on a flooding approach [10]. Nodes can learn the energy level of neighbor's and required transmission energy from neighboring nodes. The framework maintains routes to other nodes in the network on per on demand node basis. Before execute the sequence of transmitting packet data, a sender node modifies its packets header to request energy level of its receivers. On the other side, if the receiver node determines that its energy level is above a certain threshold, the packets will be successfully received. If the Link layer (of receiver node) receives an error-free packet without any duplication, it passes the packet to the Interface layer and so on. The receiver nodes will respond these packets back to sender. After receiving a respond packet from neighbors, node can then use information contained in the reply packet (i.e. the energy level at which the packet was received) to compute the most capable neighbor whose adequate energy to receive transmitted packets in order to reach the next destination node. Nodes can learn the optimum transmission with sufficient energy metric toward neighboring nodes. In the next step, sender node may invoke a propagation package to transmit data.

The framework comprises core algorithms that support topology development, maintenance and routing, as explained in Section 3. The topology development algorithm creates information about the connectivity tree map of current neighboring nodes. This algorithm also performs route optimization through the relay node that would result in optimum data transmission using combination metric of energy and distance values. If this is the case, the node is adjusted to become a potential relay, and involved in transmission path to the destination nodes and creates/update appropriate entries in its routing table. The packet is then processed by the layering modules with the result that one of the following actions is taken: (i) the packet is passed to the subsequent layers if both MAC and IP addresses match; (ii) the packet is dropped if neither MAC nor IP addresses match; or (iii) the packet is forwarded to another node when only the MAC address matches. In the latter case, nodes search the routing table to find the next route (relay) node with the higher energy and adjust the transmission energy needed to reach next destination node. When receiver receives encapsulated packet data from the higher layers it searches the routing table to see if a route toward the destination node exists. If this is not the case, node searches the neighbor table to see if information regarding the destination node is available. If this is not the case, the packets will be discarded and node transmits the route maintenance packet with energy metric contained to its neighborhood. After the neighbor node replies with a packet of its own then route optimization follows as described previously. When nodes are mobile and no data packets are available for transmission, a source node required to transmit explicit maintenance packets periodically to maintain a topology. The role of the route maintenance algorithm is to ensure that connectivity to neighbors is taken care and a minimum flow of packets is transmitted in order to maintain the route when there are no data packets available to be transmitted.

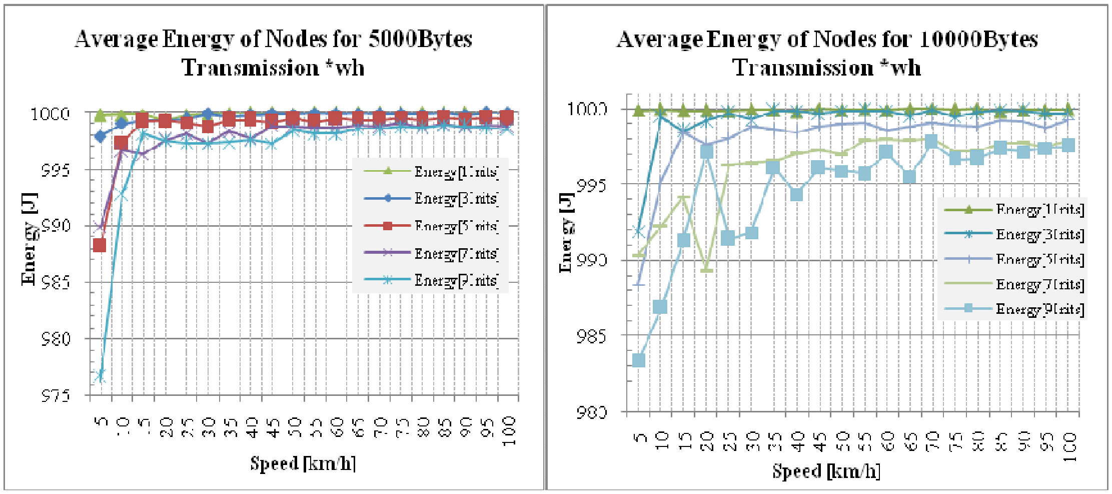
5. PERFORMANCE EVALUATION

In this section, an evaluation of the framework is discussed and followed by a number of performance issues associated with energy metric and route maintenance.

Energy Performance

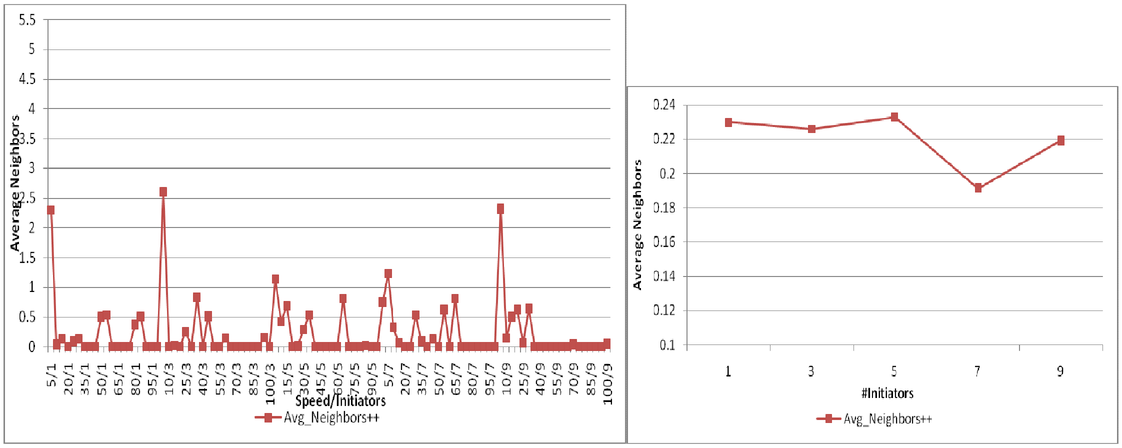
As discussed in Section 3, the more densely nodes population in the network the higher the average number of potential relay nodes, the busier communication among nodes, and the more average energy consumed in the network. The simulation topology consists of a 500x500 m^2 area network with 10 randomly positioned dynamic nodes and different number of source nodes of 1, 3, 5, 7, 9 nodes for each experiment. At initial stage, nodes have energy of 1000 J. Beside existing default overhead packets, simulation conducts source node to transmit packet's which contain data with size of 5000 and 10000 bytes. Source node consecutively transmits packets while moving during simulation. In order to achieve a reasonably high delivery ratio of dynamic network with moving nodes, average speed of network incrementally set at 5 km/h. The simulation trace captured network behavior is executed at each end of source's events in every 100 cycle.

Figure 2 shows the average energy necessary to transmit all data packets versus the average speeds of nodes in the network topology of 10 nodes. When average speed of network increases in Figure 2, average energy remained in each node is similar to its initial level. The speeder the node's move the higher rate of topology change, thus the lesser number of average neighbors and the lower successful transmission, as shown in Figure 3.



(a) (b)

FIGURE 2: The average energy needed to transmit all data packets versus the average speed of nodes in the network with 10 nodes.



(a) (b)

FIGURE 3: Average neighbors at different speed and different number of Initiators.

Figure 4 shows the average time necessary to transmit all data packets versus the average speeds of nodes in the network topology of 10 nodes. Figure 4 also indicates the average number of times packets are forwarded before reaching its destination node, i.e., average number of relay route (hop). This number of relays is dependent on the number of nodes and nodes density. The more nodes in the network then the higher the probability of having more relay nodes between source and final destination nodes. We observe that the aggregate transmission energy increases as the number of relay nodes increases. At first the aggregate transmission energy decreases rapidly when there are between an average of 3 and 4 relay nodes present.

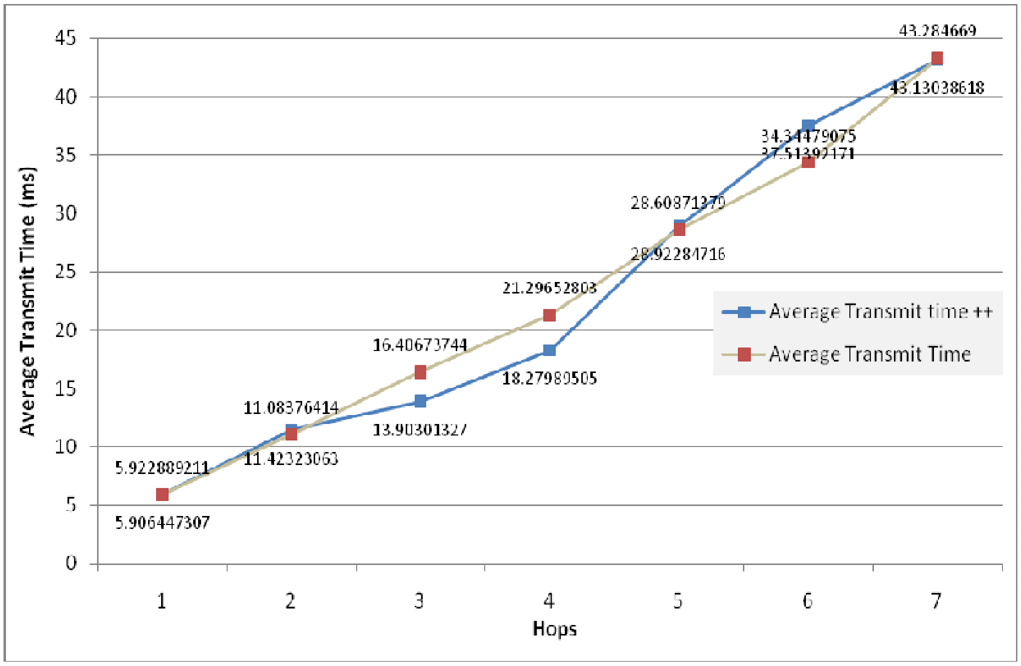
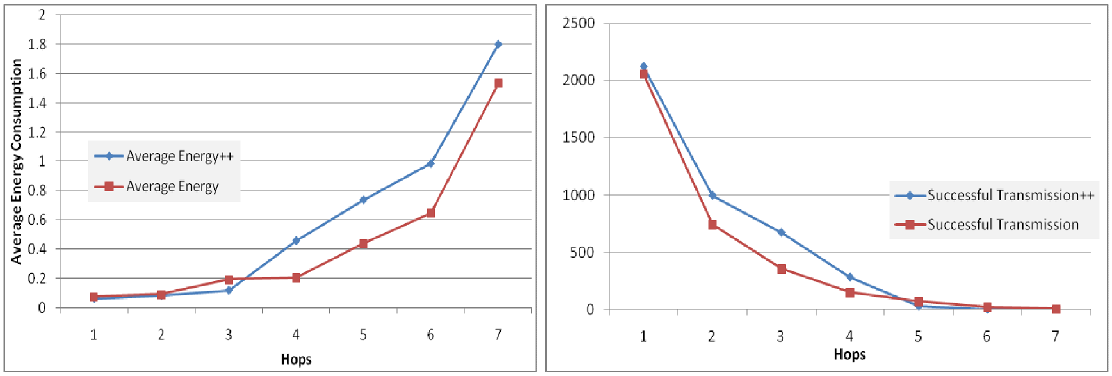


FIGURE 4: The average time needed to transmit all data packets in the network with 10 nodes with 10000bytes packet data.

Figure 5 shows the chart in terms of energy level alone. The figure draws out the result that it should not pay to have more than four relay nodes per source-destination pair for network with 10 nodes. Having more than four relay nodes may increase energy consumption and end-to-end delay. Figure 5 also indicates that the higher successful transmission was obtained for condition with lesser relay nodes were added between source-destination pairs. Comparing the two scenarios in Figure 5, we clearly observe the benefit of adding relay nodes. However, even if no relay nodes are found between source-destination pairs, by default node will keep the closest relay node to maintain adequate energy level to communicate with a destination node.



(a) (b)

FIGURE 5: (a) The average energy required to transmit all data packets and (b) the successful transmission in the network with 10 nodes with 10000bytes packet data, with the framework (SuccessfullTransmission++) and without the framework (SuccessfullTransmission).

Connectivity Maintenance

In this section, we analyze the performance of the framework in support of maintaining connectivity of mobile nodes using the charts previously shown and Figure 6. Figure 6 shows the total successful transmission per hop, while Figure 5 shows the transmission success ratio versus the number of hops and Figure 4 shows the corresponding average transmission time for each hop. We define the “transmission success ratio” as the number of fragmented packets that are fully received by the corresponding final destination nodes. The simulation includes 10 nodes in a 500x500 m^2 area network. Source node is chosen semi randomly and incrementally starting from one, three, five, and nine nodes to transmit a TCP packets flow to randomly chosen final destination nodes. Each iteration cycle consists of 10000 bytes packets which transmitted using different time intervals. From Figure 4, Figure 5, and Figure 6, we highlight concluded facts as follows. Fact (I): Nodes operating in low speed mode below 30 km/h. As a result, relay nodes remain in the path of a route for longer periods which translates into low route change updates. This condition results in a high transmission success ratio, even in the case of transmission of big packets size between source and final destination pairs. Fact (II): Nodes operating in faster mode, between 30 km/h and 60 km/h. The relay node changes frequently to a different location and being adjusted to take appropriate measures. As a result, the transmission success ratio is still high even for the case where nodes move faster. Fact (III): Nodes operating in high speed mode. Because of high mobility, most of the routes change and some nodes are outside of coverage distance. However, packets are not transmitted at a high successful rate to maintain routes in the network due to the long silence-intervals between packets. Data packets that are transmitted by nodes located in such situation are likely to be lost. This is because sender nodes may not have accurate information concerning the next hop route. As a result, the transmission success ratio is low.

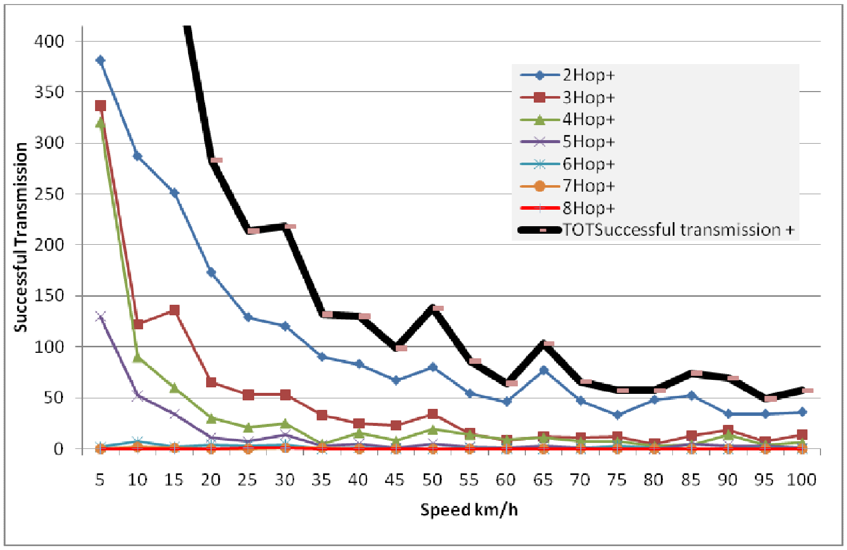


FIGURE 6: The relationship among successful transmission, speed, and average transmission time for network with 10 nodes with 10000bytes packet data.

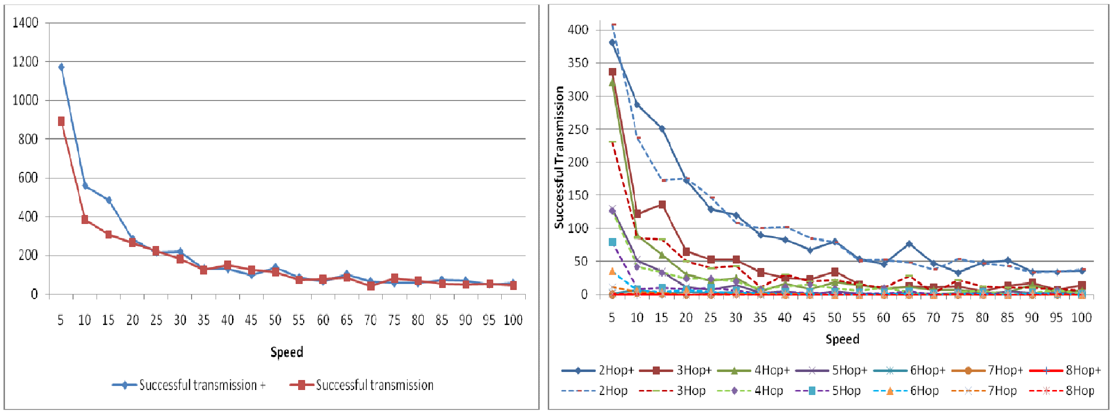
Determining the optimum value of the blank route path appearance to overcome high speed mobility nodes (in order to guarantee a certain success ratio) is a complex issue. This value is dependent on the size of the network and the node density as well as mobility and data packet inter-arrival rate. Maintaining a route with fewer relays will requires less signaling packets both in terms of topology maintenance and routing-maintenance packets. On the contrary, larger areas with high density nodes will likely support routes and maintain connectivity with several relays.

6. DISCUSSION

The proposed framework assumes that nodes are capable of dynamically adjusting their relay nodes on per move step base. It attempts to minimize the number of relay nodes between source and final destination pairs and at the same time maintain the node's energy level required. This behavior is almost similar to MANET routing protocols (e.g., AODV, DSR and TORA). One common property of these routing protocols is that they discover routes using a variety of broadcast flooding protocols by transmitting at maximum power in order to minimize the number of relay nodes between any source and final destination pair. MANET routing protocols do not provide a suitable mechanism for discovering optimum energy aware routes in wireless ad hoc networks and at the same time keep moderate the number of hops. Delivering data packets in wireless ad hoc networks using minimum-hop routes, however, requires more transmission power to reach destinations.

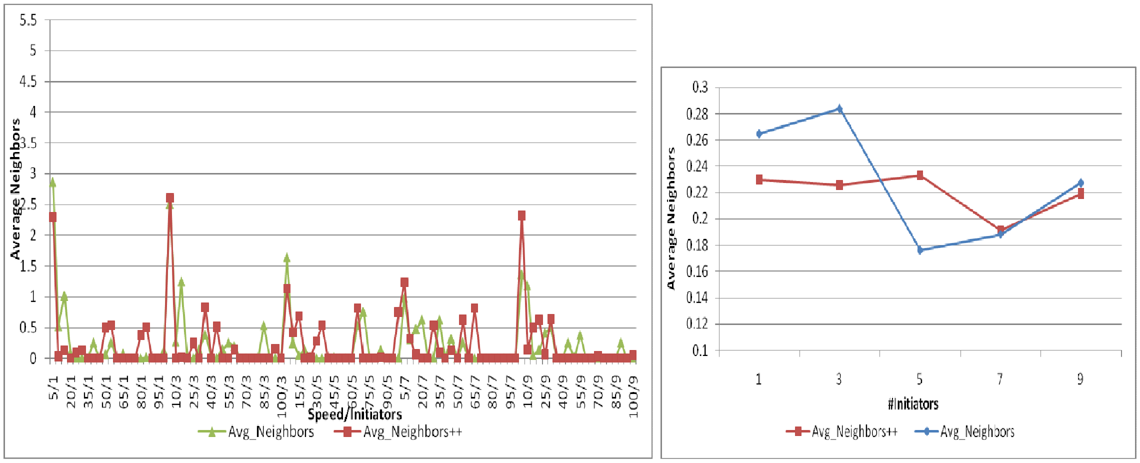
The framework discovers a balance between energy and routes on-demand on a propagation node-to-node basis. A different approach would generate full routing tables in advance where, all nodes in the network would be aware of energy level and distance routes to all other nodes in the network. Such protocol behavior is similar to Link State Routing (LSR) using energy level metric as the link cost unit. The basic LSR operation requires that each node in the network to broadcast a routing packet. The broadcast packet contains information about the combination of energy metric and distance metric of all known destinations. After collecting packets from all parts of the network, any node should be capable of computing optimum routes to other nodes.

We compare the proposed framework and similar Link State Routing network to best understand the various tradeoffs and limitations of design. We consider a network composed of N nodes located within transmission range of each other. Similar Link State Routing network can compute the optimum energy metric with higher energy level ratio and shortest distance to a next relay node by listening to a reply topology development and topology maintenance packets transmitted by the neighbors. These packets include the energy level available and other information to transmit the packet. The received packet may require to be forwarded by other nodes and then propagate them again to the entire network. Each node computes routes to any other node in the network using a standard link-state Dijkstra algorithm. The iteration of propagation events to be entirely flooded mainly depends on the density of nodes in the network.



(a) (b)

FIGURE 7: Comparison of successful data transmission between The Framework (Successful transmission+; Hop+) and Link State Routing (Successful transmission; Hop).
(a). There is about 18% differences.
(b) Successful data transmission each hops at different speed.



(a) (b)

FIGURE 8: Average neighbors at different speed and different number of Initiators for comparison between the framework (Avg_Neighbors++) and similar LSR (Avg_Neighbors).

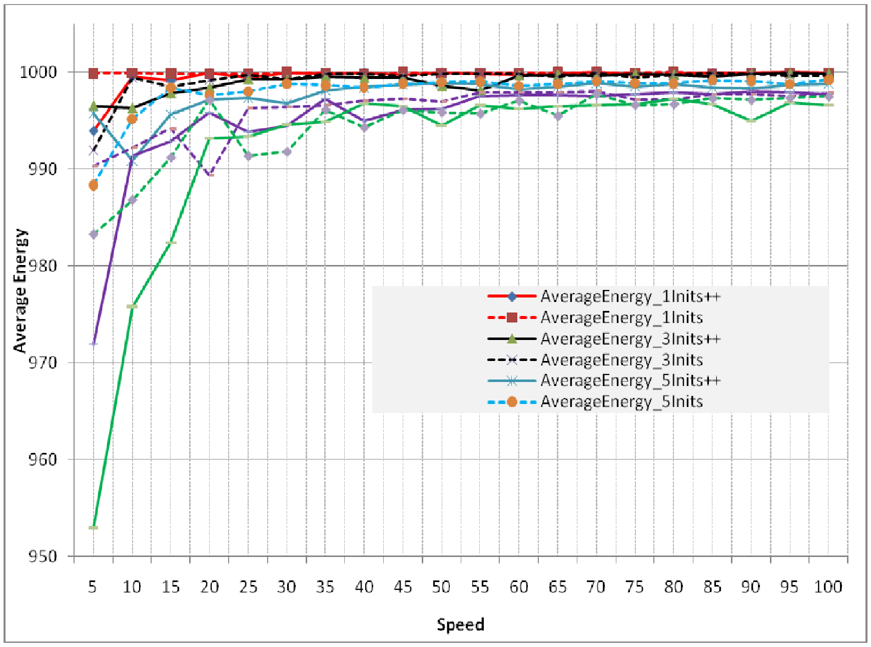


FIGURE 9: Aggregated Energy Level Ratio Consumed by Data and Signaling for The Framework (AverageEnergy++) and Link State Routing (AverageEnergy).

Figure 9 shows a simulation trace of the aggregate energy level ratio consumed by both overhead and data packets for both the framework and Link State Routing. The network simulation consists of 10 static nodes a 500x500 in size with TCP flows transmitting a 5000, and 10000-byte packet. In the case of Link State Routing, topology construction packets are first transmitted to generate full routing tables. Once routing information is available, in Link State Routing, packets data are transmitted. In the case of the proposed framework, packets data are first transmitted at high power because destination nodes is unknown to source nodes and several neighbor nodes are potentially to be selected as relay nodes. Figure 9 shows the remained energy offset to converge to optimum routes for both the framework and Link State Routing. It can be observed from Figure 9 that relative to the remained energy level consumed by signaling packets and the packet contain data, the contribution of data transmission to the overall energy level/consumption is about 2.38%. This result implies an important design principle for future energy-connectivity aware routing protocols is important in data transmissions at mobile environment.

Many researchers have proposed protocols to manage energy consumption in MANET. Among existing protocols, we can group as two categories: topology control oriented and broadcast oriented. The proposed framework could be included in broadcast oriented part. The first category (topology control oriented) assigns the transmission power for each node such that the network is connected independently of broadcast utilization. That means that all nodes can be a source of a broadcast and are able to reach all nodes of the network. The second category (broadcast oriented) considers the broadcast process from a given source node by means of the minimum-energy broadcast tree. It has condition that the source can reach every node of the network. Research e.g. [4] describes a localized protocol where each node requires only the knowledge of its distance to all neighboring nodes and distances between its neighboring nodes (or, alternatively, geographic position of itself and its neighboring nodes), while our proposed framework optimized the broadcast mechanism by means of energy level and distance metrics. Several types of broadcast are introduced to develop the topology, to maintain the connectivity, and to adjust the energy level required for transmission in the network.

7. CONSLUSION & FUTURE WORK

In this paper, we have presented the generic framework of an energy level and distance aware routing optimization for wireless ad hoc networks. We evaluated this framework and compared its performance to similar Link State Routing network. We found that the framework is able to balance of both remained energy level ratio and distance metric routes to maintain connectivity compared to similar Link State Routing with its point to point on-demand design.

8. REFERENCES

1. Benjie Chen, Kyle Jamieson, Hari Balakrishnan, Robert Morris, Span. "*An energy-efficient coordination algorithm for topology maintenance in Ad Hoc wireless networks*". Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy, July 2001, p.85-96.

2. Chang, J., and Tassiulas, L. "*Energy Conserving Routing in Wireless Ad Hoc Networks*". In Proceedings of IEEE INFOCOM, Tel Aviv, Israel, 2000.

3. Chang-Woo Ahn, Sang-Hwa Chung, Tae-Hun Kim, Su-Young Kang. *"A Node-Disjoint Multipath Routing Protocol Based on AODV in Mobile Adhoc Networks"*. In: Proceeding of Seventh International Conference of Information Technology ITNG2010: 828-833, April 2010.

4. F. Ingelrest, D. Simplot-Ryl. "*Localized broadcast incremental power protocol for wireless ad hoc networks*". Proceedings of the 10th IEEE Symposium on Computers and Communications, Cartagena, Spain, 2005, pp. 28–33.

5. Feeney, L., and Nilsson, M. "*Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment*". In Proceedings of IEEE INFOCOM, Anchorage, AK, 2001.

6. Fenner, W. "*Internet Group Management Protocol*", Version 2, Nov 1997. RFC-2236.

7. Heinzelman, W., Chandrakasan, A., Balakrishnan, H. *"Energy-efficient communication protocol for wireless microsensor networks"*. In: Proceedings of the 33rd International Conference on System Sciences (HICSS): 1–10, 2000.

8. IEEE Std. 802.11-1999, Part11. *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications"*. IEEE Std. 802.11, (1999).

9. J. Wu and F. Dai. "*Broadcasting In Ad Hoc Networks Based On Self-Pruning*". Int'l J. Foundations of Computer Science, vol. 14, no. 2, Apr. 2003, pp. 201-221.

10. J. Wu and F. Dai. "*Efficient broadcasting with guaranteed coverage in mobile ad hoc networks*". IEEE Trans. Mobile Comput. 4 (3) (2005), pp. 1–12.

11. J. Wu and F. Dai. "*Mobility-sensitive topology control in mobile ad hoc networks*". IEEE Trans. Parallel Distributed Syst. 17 (6) (2006), pp. 522–535.

12. Javier G., Andrew T. C., Mahmoud N., Chatschik B. *"Conserving Transmission Power in Wireless Ad Hoc Networks"*. Network Protocols Ninth International Conference on ICNP: 24-34, Nov 2001.

13. Julien Cartigny, David Simplot , Ivan Stojmenovic. "*Localized Minimum-energy broadcasting in ad-hoc networks*". Twenty-Second Annual Conference of the IEEE Computer and Communication society, INFOCOM 2003.

14. Masato, Tsuru. *"Simulation-based Evaluation of TCP Performance on Wireless Networks"*. Journal of the Japan Society for Simulation Technology: 67-73, 2009.

15. Moreno M.T., C. Steven, E. Felix Schmidt, Hartenstein H. *"IEEE 802.11-based one-hop broadcast communications: understanding transmission success and failure under different radio propagation environments"*. In: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems, SESSION: Modeling and performance evaluation II (physical and link layer), ISBN:1-59593-477-4: 68 – 77, 2006.

16. N. Bansal, Z. Liu. "*Capacity, delay and mobility in wireless ad-hoc networks*". INFOCOM, San Franciso, USA, 2003, pp. 1553–1563.

17. Nozomu, Nemoto. *"Consideration and Evaluation of Wireless Mesh Network"*. Nomura Research Institute (NRI) Pacific Advanced Technologies Eng.: 70-85, 2006.

18. Prasanthi. S, Sang-Hwa Chung. *"An Efficient Algorithm for the Performance of TCP over Multi-hop Wireless Mesh Networks"*. In: Proceeding of Seventh International Conference of Information Technology ITNG2010: 816-821, April 2010.

19. Robin Kravets, P. Krishnan. "*Power management techniques for mobile communication*". Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, p.157-168, Dallas, Texas, United States, October 25-30, 1998.

20. Rodoplu, V., and Meng, T. H. "*Minimum Energy Mobile Wireless Networks*". In Proceedings of the IEEE International Conference on Communications (ICC), Atlanta, GA, vol. 3, pp. 1633-1639, June 1998.

21. T. Camp, J. Boleng and V. Davies. "*A survey of mobility models for ad hoc network research*" Wireless Commun. Mobile Computing, special issue on mobile ad hoc networking: research, trends and applications 2 (5) (2002), pp. 483–502.

22. Y.C. Tseng, S.Y. Ni, Y.S. Chen, and J.P. Sheu. *"The broadcast storm problem in a mobile ad hoc network"*. Wireless Networks, 8(2/3):153–167, Mar-May 2002.

23. Ya Xu , John Heidemann , Deborah Estrin. "*Geography-informed energy conservation for Ad Hoc routing*". Proceedings of the 7th annual international conference on Mobile computing and networking, Rome, Italy, p.70-84, July 2001.

SDC: A Distributed Clustering Protocol

Yan Li yan.li@engr.uconn.edu
Computer Science and Engineering
University of Connecticut
Storrs, CT 06269, USA

Li Lao llao@google.com
Google Santa Monica
Santa Monica, CA 90401, USA

Jun-Hong Cui jcui@engr.uconn.edu
Computer Science and Engineering
University of Connecticut
Storrs, CT 06269, USA

Abstract

Network clustering is an important technique used in many large-scale distributed systems. Given good design and implementation, network clustering can significantly enhance the system's scalability and efficiency. However, it is very challenging to design a good clustering protocol for networks that scale fast and change continuously. In this paper, we propose a distributed network clustering protocol SDC targeting large-scale decentralized systems. In SDC, clusters are dynamically formed and adjusted based on SCM, a practical clustering accuracy measure. Based on SCM, each node can join or leave a cluster such that the clustering accuracy of the whole network can be improved. One big advantage of SDC is it can recover accurate clusters from node dynamics with very low message overhead. Through extensive simulations, we conclude that SDC is able to discover good quality clusters very efficiently.

Keywords: network clustering, distributed algorithm, Scaled Coverage Measure, SDC, dynamic network

1. INTRODUCTION

Clustering is an important technique studied in various areas, such as biology, chemistry, linguistics, physics, and sociology. The basic goal of clustering is to group data in such a way that data in the same cluster shares certain similarity. In this paper, we study one interesting type of clustering: *network clustering*, which partitions a network topology into clusters so that nodes in the same clusters are highly connected and between clusters are sparsely connected.

Network clustering has become an important technique in different networking research areas. With a good network clustering algorithm, we can design scalable and efficient routing protocols [13] [3] [1] [23], enhance scalability and efficiency of large-scale distributed systems [16] [2] [21] [9], and resolve many critical networking issues such as virus spreading [26] [15] [32], QoS [19] [18], network robustness [6] [4] [12] [25], to name a few. In [22], network clustering is used to study the clustering features of the AS-level Internet topology and a realistic topology model is designed based on the observed clustering features.

Network clustering can be performed in both centralized and distributed ways. Centralized network clustering is an off-line procedure, in which complete network topology information is required. Thus, centralized clustering is usually used for small networks or off-line data analysis. In our work, we focus on distributed clustering techniques, which are designed for large-scale distributed systems.

To design a good network clustering protocol, we must consider the following design criteria. First of all, as a natural requirement of network clustering, nodes in the same clusters should be highly connected, and less connected between clusters. Secondly, a good clustering protocol should control cluster size (or cluster diameter) well. Thirdly, the number of “orphan” nodes should be minimized. Lastly, a good distributed clustering protocol should take node dynamics into account, especially when the clustering targets are highly dynamic with frequent node entry and exit. We provide a detailed discussion on clustering criteria in Section 2.

In the literature, there has been considerable research effort addressing the problem of network clustering, but very few of them studied the problem in the scenario of large-scale distributed networks. Among the existing approaches, MCL [28] is well accepted as an efficient and accurate network clustering algorithm. However, this approach assumes that complete network topology is available at one central point, which makes it difficult to apply MCL into distributed systems. CDC [27], on the other hand, is a distributed algorithm. It forms clusters based on node connectivity. The main issue with this algorithm is that it can not handle node dynamics in a decent way: a large number of messages must be exchanged to keep accurate clusters.

With these problems in mind, we design a novel network clustering protocol: **SCM-based Distributed Clustering (SDC)**, which satisfies all the design criteria mentioned above. In SDC, clusters are dynamically formed and adjusted based on a practical clustering accuracy metric, Scaled Coverage Measure (SCM) [29]. In SDC, each network node makes its own decision to join or leave a cluster whenever clustering accuracy can be improved. To control cluster size, TTL (Time-To-Live) is piggybacked in exchanged messages to guarantee cluster diameter does not exceed a predefined threshold. SDC is a fully decentralized protocol which requires only neighbor information, and it can handle node dynamics with small message overhead while keeping good quality of clusters. Besides the basic protocol design, we also address some difficulties in scenarios of distributed networks. A common and critical issue addressed in this paper is deadlock, which is caused by simultaneous node actions. We provide some strategies to avoid and resolve deadlock conditions and analyze their effects on the performance of the protocol. Through extensive simulations, we can conclude that our proposed protocol, SDC, is able to discover high quality clusters in a very efficient way.

The rest of this paper is structured as follows. In Section 2, we introduce the background of network clustering and review some related work. In Section 3, we discuss an important concept, SCM which is the basis of our design. Then in Section 4, we present our clustering protocol SDC in detail. We show the performance of SDC by extensive simulations in Section 5. At the end of this paper, we conclude our work in Section 6.

2. BACKGROUND AND RELATED WORK

In this section, we formulate the network clustering problem and introduce a set of criteria for desired clustering approach. Then we review several existing clustering methods.

2.1 Network Model

A targeting network to be clustered can be presented as a connected, undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges connecting network nodes. Let $|V| = n$ and $|E| = m$. Then a partition $C = \{C_1, C_2, \dots C_l, \}$ of V is named as a clustering C of graph G, and C_i represents the i^{th} cluster. Each cluster must be a non-empty subset of V. Clearly, $\bigcup_{i=1}^{l} C_i = V$. The diameter of a cluster C_i is defined as the maximum length of the shortest paths between all pairs of nodes in C_i. Accordingly, if a cluster has one node, its diameter is 0. We call the clusters with diameter equal to 0 as orphan nodes.

Another metric associated with a cluster is cluster size that is defined as the number of nodes in a cluster. Cluster size and cluster diameter are closely related. In most scenarios, “control cluster size” is equivalent to “control cluster diameter”. We only differentiate these two metrics in the protocol description.

2.2 Criteria of Clustering

The network clustering problem can be formulated as finding a “good” clustering C in G such that C can accurately describe the natural clustering features in the topology. More specifically, in a “good” partition C, the intra-cluster node connectivity should be maximized and the inter-cluster node connectivity should be minimized. Therefore, node connectivity is a basic criterion to be considered in network clustering design. In addition to node connectivity, cluster size is another important metric. In large-scale distributed networks, due to the lack of knowledge about network structure, it is expensive to maintain expanded clusters. In other words, cluster diameters should be carefully controlled. A good network clustering algorithm should also take the number of orphan nodes into consideration. In most scenarios, orphan nodes are not preferred as they violate the goal of clustering and should be eliminated.

As discussed above, node connectivity, cluster diameter, and orphan nodes are important criteria for good network clustering algorithms. However, more issues need to be addressed when we cluster large-scale distributed systems. In such networks, a node only has the knowledge about its neighbors and may join or leave the network at any moment. To obtain a complete view of the network structure, a huge number of messages need to be exchanged to collect the topology information. Moreover, the obtained topology may expire very soon due to node dynamics. Re-collecting topology information on each node-entry and node-exit will overload the network with a huge amount of exchanged messages. Therefore, it is not feasible to maintain a complete and up-to-date topology in such networks. Given these concerns, a good clustering protocol for large-scale distributed systems should form clusters in a fully distributed fashion, i.e., nodes should form clusters automatically without the requirement of complete network topology, and it should be able to recover accurate clusters from node dynamics with small overhead in term of the number of exchanged messages.

2.3 Related Work

Significant research efforts have been devoted into design of network clustering methods [5] [10] [14] [17] [28] [27]. However, many existing clustering algorithms assume that the complete network topology is available at a central point. One typical research line tries to solve the MINIMUM k-CLUSTERING problem which is formulated as follows: Given a network topology G and an integer k, find a partition of G into a smallest number of l subsets so that the diameter of each subset is at most k. The MINIMUM k-CLUSTERING is proved to be NP-Complete in simple and undirected graphs [10], so most of research efforts are focused on special types of topologies. One representative work is presented in [10], which proposes a polynomial time approximation algorithm, DDP, for graphs with dominating diametral path. DDP does not form clusters based on node connectivity, so it can not guarantee accurate clustering.

The Markov Cluster (MCL) [28] is a connectivity-based centralized network clustering algorithm. The basic idea behind this algorithm is flow simulation. In this algorithm, an input graph G is mapped in a generic way onto a Markov matrix. Then the set of transition probabilities are iteratively recomputed via matrix expansion and inflation. An infinite sequence consisting of repeated alternation of expansion and inflation constitutes a new algebraic process called the Markov Cluster (MCL) process. The heuristic behind this algorithm is that a flow between sparsely connected dense regions evaporates after MCL process. Therefore, it is easy to detect dense regions in the original graph which are the output clusters. MCL algorithm can achieve high clustering accuracy. However, due to its centralized feature, it can not be used in distributed systems.

There have been many proposals for network clustering in large-scale decentralized systems [8, 11, 20, 30, 31]. Among existing decentralized clustering algorithms, one representative work is [27], a connectivity based distributed network clustering algorithm, CDC, which is designed for p2p networks. In CDC, a set of peers are selected as "originators" and clusters are discovered around these peers by TTL-controlled message flooding. If the "originators" are well distributed in the network, clusters with good quality can be formed. The CDC scheme is a fully distributed approach and the cluster size can be effectively controlled by TTL. The main issue of CDC is the selection of "originators" which can affect the accuracy of clustering significantly. So far, there is no good solution to well distribute "originators". Thus, the clustering accuracy can not be guaranteed. Another issue with CDC is it can not efficiently handle node dynamics. To maintain good clustering quality, the whole network has to be re-clustered at each node join or leave, which introduces a huge amount of message overhead.

In summary, there is no existing clustering method which can satisfy all the criteria for network clustering in large-scale distributed systems. In this paper, we propose a novel network clustering protocol, SDC. It is fully distributed and can form high quality clusters in highly dynamic systems with small message overhead.

3. SCALED COVERAGE MEASURE

Before introducing our protocol, we first discuss Scaled Coverage Measure (SCM), a practical metric to evaluate the accuracy of connectivity-based clustering algorithms proposed by S.Van Dangon [28].

We assume $C = \{C_1, C_2, \dots C_l,\}$ is a clustering on network $G = (V,E)$. Given a node $v_i \in V$, we have the following notations:

- **Nbr(**v_i**)** is the set of neighbors of node v_i;
- **Clust**(v_i) is the set of nodes in the same cluster as node v_i (excluding v_i);
- **FalsePos**(v_i, C) is the set of nodes in the same cluster as v_i but not neighbors of v_i;
- **FalseNeg**(v_i, C) is the set of neighbors of v_i but not in the same cluster as v_i;

Then the Scaled Coverage Measure of node v_i, $SCM(v_i)$, is defined as:

$$1 - \frac{|FalsePos(v_i, C)| + \|FalseNeg(v_i, C)\|}{|Nbr(v_i) \cup Clust(v_i)|}.(1)$$

For graph G, the SCM value, $SCM(G)$, is defined as the average of the SCM values of all the nodes, that is, $SCM(G) = (\sum_{v_i} SCM(v_i))/n$, which lies in [0, 1].

SCM well reflects the significance of clustering features in a given network topology. First of all, it is easy to see that the higher the SCM, the smaller the connectivity between clusters and the higher the connectivity within clusters. For graphs containing only isolated clusters/subgraphs that are themselves fully connected, the SCM value is 1. Secondly, for any graph, there exists a highest SCM value which is determined solely by the network structure. If the network does not contain significant clustering substructures, this highest "available" SCM value can be very small. However, if we evaluate two clustering techniques on the same network, the one which results in a higher SCM value discovers more accurate clustering substructures than the one with smaller SCM value, although both resultant SCM values could be very small. Lastly, the SCM value of an orphan node is 0, which matches our goal of minimizing the number of orphan nodes.

Based on the definition of SCM, the network clustering problem can be simplified as partitioning a network topology so that its SCM is maximized. Our proposed SDC protocol exactly follows this idea, adaptively forming clusters in an aggressive manner.

Simplified Notations To simplify the computation in SDC, we can re-express SCM at node v_i as follows:

$$1 - \frac{b_{v_i}}{a_{v_i}}, (2)$$

Thus,

$$b_{v_i} = |FalsePos(v_i, C)| + |FalseNeg(v_i, C)|,$$

and

$$a_{v_i} = |Nbr(v_i) \bigcup Clust(v_i)|.$$

If node v_i is an orphan node, $b_{v_i} = a_{v_i} = deg\ (v_i)$, where $deg\ (v_i)$ is the degree of node v_i. These two parameters b_{v_i} and a_{v_i} can be easily updated based on neighbor information upon node joining and leaving the cluster.

In the next section, we show how SCM is utilized in the SDC protocol to form clusters in a distributed fashion.

4. SCM-BASED DISTRIBUTED CLUSTERING: SDC

SDC is a fully distributed clustering protocol. A node in the network maintains only its own state information: 1) the cluster it currently belongs to, identified by a unique id $clust\_id$, 2) the current number of nodes in its cluster $clust\_size$, 3) the two parameters for SCM computation b_{v_i} and a_{v_i}.

To cluster a network from scratch, every node is initialized as an orphan node with its own $clust\_id$ (any unique identifier) and $clust\_size$ that is equal to 1. For any node v_i, the two parameters for SCM computation, b_{v_i} and a_{v_i}, are initialized as $deg\ (v_i)$. Then every node starts its own clustering procedure independently at a random time by sending requests to its neighbors. If the requests are accepted by the neighbors, the clustering procedure continues with a few rounds of message exchange until the node joins a cluster. Otherwise, the node can select to serve its neighbor's requests or start a new round of clustering. Clustering of individual node is an independent and local procedure. The clustering of the whole network ends when no message is exchanged. Next, we describe the protocol in detail.

4.1 Protocol Description

In SDC, nodes form clusters in a greedy manner based on SCM. Each node tries to cluster with a subset of neighbors which leads to a higher SCM value than cluster with the other neighbors. When a node is actively involved in a clustering procedure, it is either in "Clustering" mode, i.e. it starts the clustering procedure, or in "Serving" mode as it is serving another node's clustering, but not both. The clustering procedure of any node, v_i, involves the exchange of a set of messages elaborated as follows.

- $\mathbf{Clust\_Request}$. A node v_i starts its own clustering procedure by broadcast $Clust\_Request$ message to all the neighbors. Once the message is sent out, v_i is in the "Clustering" mode and waits for the response from its neighbors. A timer is set up to control the waiting time. During this waiting period, v_i can not accept and serve the clustering requests from other nodes. It buffers all the received requests and handles them after the current clustering. If all the neighbors response before timer expires, v_i confirms all the replies and the clustering procedure continues. If the timer expires and no response is received, v_i checks the buffered requests and selects one to serve. If the buffer is empty, v_i restarts its clustering procedure after a small random time period.
- $\mathbf{Clust\_Reply}$. Upon receiving $Clust\_Request$ from the neighbor v_i, node v_j sends back a $Clust\_Reply$ message if it is not actively involved in any clustering procedure. Otherwise, it refuses the clustering request by sending a $Clust\_Refuse$ message back to v_i as explained next. By sending out a $Clust\_Reply$, v_j claims its willingness to serve the clustering request of v_i and waits for confirmation from v_i. The $Clust\_Reply$ message

carries the current cluster information of v_j such as $clust\_id$, $clust\_size$ and $\Delta SCM(v_j)$ that shows the gain in $SCM(v_j)$ assuming node v_i joins the cluster of v_j if it is not in $Clust(v_j)$ or leaves $Clust(v_j)$ otherwise. The computation of $\Delta SCM(v_j)$ only requires the knowledge of whether v_i and v_j are directed connected. Specifically, let us consider the case that v_j is in a different cluster from v_i. If v_j is a neighbor of v_i,

$$\Delta SCM(v_j) = \frac{1}{a_{v_j}}. (3)$$

On the other hand, if v_j is not directly connected with v_i,

$$\Delta SCM(v_j) = \frac{b_{v_j}}{a_{v_j}} - \frac{b_{v_j}+1}{a_{v_j}+1}. (4)$$

The gain in SCM for nodes in the same cluster of v_i is computed in a slightly different way and can be easily derived.

- **Service_Confirm**. After receives replies from its neighbors, a requesting node needs to confirm the acceptance of the service provided by the neighbors. A $Service\_Confirm$ is sent back to the neighbor for this purpose. Once receives $Service\_Confirm$, a node can not serve others or request clustering for itself.
- **Clust_Lock**. A node that decides to serve a neighbor's clustering may receive $Clust\_Reply$ from its neighbors for its previous requests. This situation happens if a node requests for clustering service but none of it's neighbors are available to serve it at the moment. Those neighbors will buffer the request as mentioned above. After a while, some of the neighbors may become available again and try to serve the buffered requests. When a node that is in the serving mode receives a $Clust\_Reply$ for its previous requests, it replies with a $Clust\_Lock$ message which indicates it is serving others and can not accept the service offer.
- **Clust_Refuse**. If a node is in either “Clustering” or ”Serving” mode when it receives a $Clust\_Request$ message, it informs the requester its unavailability for the request by sending a $Clust\_Refuse$ message. If a node is refused by all the neighbors, it either serves one buffered request if there is any or re-starts a new round of clustering procedure after a small random time interval.
- **Clust_Confirm**. If node v_i receives a $Clust\_Reply$ from neighbor v_j during the waiting period after it broadcasts a request, it confirms the service offer by sending back v_j a $Clust\_Confirm$ message and starts waiting for more $Clust\_Reply$ messages from the other nodes in $Clust(v_j)$. When node v_j receives the $Clust\_Confirm$, it enters the “Serving” mode and forwards the request message to all the other nodes in its current cluster through flooding. When another node v_k in $Clust(v_j)$ receives the request, it sends a $Clust\_Reply$ with its own $\Delta SCM(v_k)$ back to the request node v_i.
- **Clust_Reject**. To control the cluster granularity and the number of exchanged messages, every $Clust\_Request$ message carries a TTL field. Based on the value of TTL, a node can determine whether the cluster diameter will exceed a predefined threshold after v_i joins. If the TTL expires, v_j stops forwarding $Clust\_Request$ and sends a $Clust\_Reject$ message to v_i. Once receiving $Clust\_Reject$, v_i does not take $Clust(v_j)$ as a potential cluster to join.
- **Clust_Update**. After node v_i receives $Clust\_Reply$ from all the nodes in its current cluster and the neighbor cluster C_l (in the case that no $Clust\_Reject$ is received from C_l), it computes the overall gain $\Delta SCM(G)$ based on the received information. We use Δ_{leave} and Δ_{join} for the gain in SCM as if v_i leaves its original cluster and joins C_l. Then the overall $\Delta SCM(G)$ is computed as:

$$\Delta SCM(G) = \Delta_{join} + \Delta_{leave}. (5)$$

where Δ_{join} is the sum of the gain in SCM received from all the nodes in v_i's current cluster and Δ_{leave} is the sum of the received gain in SCM from all the nodes in the neighbor cluster C_i. If $\Delta SCM > 0$, v_i should join C_i. There might be multiple clusters for which ΔSCM are positive, v_i should join the one corresponding to the maximum ΔSCM. Once v_i determines which cluster to join, a $Clust\_Update$ message containing v_i's node id and its original $clust\_id$ is flooded in its original cluster and the new cluster it is joining. Then, v_i and every node receiving $Clust\_Update$ need to update the clustering information.

After v_i joins the new cluster, its neighbors in the other clusters are affected. These nodes will check whether they should move to v_i's cluster for a higher SCM in the same way as node v_i does. The whole clustering procedure ends if there is no exchanged message.

4.2 Handling Deadlocks

A critical issue that distributed network protocols must handle is how to detect and resolve deadlocks. In SDC, nodes may enter deadlock conditions if they start the clustering request simultaneous as their neighbors. When deadlock occurs, none of them can get served as they are all waiting for each other's replies. Deadlocks can cause the involved nodes to be in a busy waiting state infinitely, so those nodes will never get clustered.

We using the following two methods to avoid and resolve deadlocks.

- State Reset: If a node is in a waiting state which can be waiting for clustering reply, service confirm, etc., it will enter the next state automatically after a certain amount of time whether or not it receives replies from its neighbors. For example, a node sends $Clust\_Request$ to its neighbors. It starts a timer right after the requests are sent. When the timer times out, the node enters the corresponding next state based on whether or not it receives any $Clust\_Reply$. The length of the timer can be estimated based on RTT and the predefined TTL.
- Randomization: A direct reason for the deadlock scenario is the simultaneous node actions. Therefore, we introduce a randomized delay for each node before it takes actions. This randomization can affect the performance of SDC significantly. A short randomized delay may not be effective to resolve the deadlock condition while long delays are more effective but can slow down the whole clustering procedure. To analyze the effects, we provide simulation evaluations in the next section.

4.3 An Example

A simple example is shown in Fig. 1 to illustrate the SDC clustering procedure. In this example, TTL is set to 2, so the diameter of any cluster will not exceed 2. In Fig. 1a, node 0 wants to be clustered with other nodes. It first sends $Clust\_Request$ messages to all of its neighbors. Each neighbor node upon receiving the $Clust\_Reply$ sends its $clust\_id$, $clust\_size$ and SCM gain back to node 0 as shown in Fig. 1b. After receiving replies from all the neighbor, node 0 sends $Clust\_Confirm$ to accept the service offer so that its requests are forwarded to every node in the clusters of A and B via flooding, as shown in Fig. 1c. At the same time, node 7 also starts its clustering procedure by sending a $Clust\_Request$ to its neighbor 2. Since node 2 is in the "Serving" mode, the request from node 7 is refused. Thus node 7 waits for a small period of time before the next clustering attempt. In clusters A and B, every node which receives $Clust\_Request$ computes its SCM gain and sends $Clust\_Reply$ back to 0 (Fig. 1d). Node 0 then computes $\Delta SCM(G)$ based on the received information and joins Cluster A at the end (Fig. 1e). Since node 4 is affected by node 0's clustering, it starts its own clustering procedure in the way as node 0 does (Fig. 1f).

4.4 Handling Node Dynamics

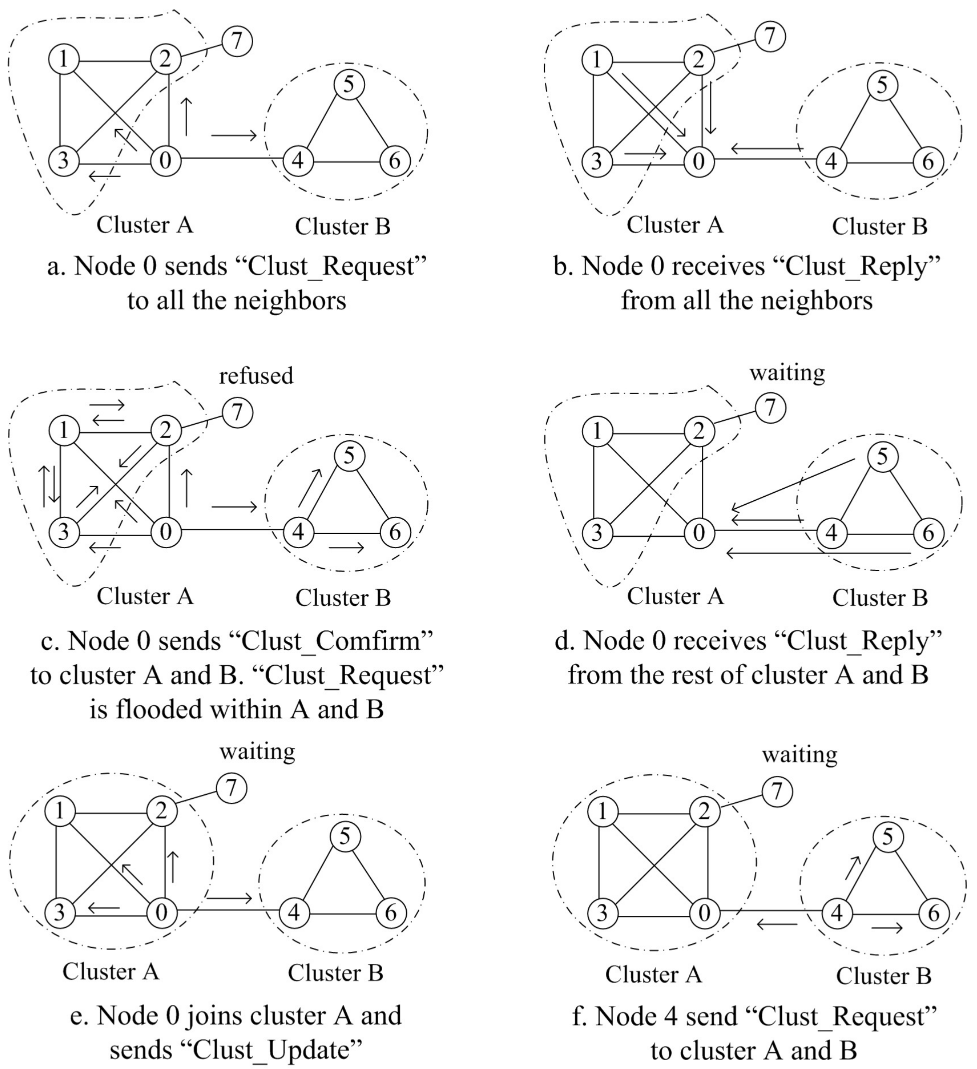
In large-scale distributed systems, node entry and exit can occur at arbitrary time and the network structure may change continuously. Node dynamics can degrade the existing clusters and must be handled by the clustering protocol. Re-do the whole clustering procedure may recover good clustering accuracy. However, it is very inefficient and the procedure may never stabilize if node entry and exit happens frequently. Therefore, designing an effective and efficient scheme to handle node dynamics is a critical demand for distributed clustering.

Our SDC protocol can naturally handle node dynamics in a decent way. Whenever a new node v_i joins the system, it is first initialized as an orphan node and gets its own $clust\_id$ and $clust\_size$ (which is 1). Since the network structure between node v_i and its neighbors is changed, a **Join** message carrying v_i's $clust\_id$ is issued by v_i to all of the neighbors so that they can update their SCM. As v_i's joining changes its neighbors' connectivity, the affected neighbor nodes should perform a new round of clustering procedure. When a node wants to leave, it sends a **Leave** message to each of its neighbors as well as every other node in its cluster through flooding so that the $clust\_size$ and SCM values of the affected nodes can be updated. This will also activate a new round of clustering procedures at these affected nodes. The logic behind this scheme comes from the fact that node entry and exit are localized events and only a few nodes are affected and need to be re-clustered.

Some overhead is introduced when SDC handles node dynamics. Nevertheless, this overhead is very small since only neighbors and/or the nodes in the same cluster are directly affected. In next section, we will show that SDC can achieve good clustering accuracy with low overhead in the presence of node dynamics. In contrast, CDC has to re-do the complete clustering procedure for any node join or leave in order to maintain good clustering accuracy, which introduces a lot of overhead.

5. SIMULATION EVALUATIONS

In this section, we conduct simulations to evaluate the performance of SDC, comparing it with the decentralized clustering scheme, CDC.



a. Node 0 sends "Clust_Request" to all the neighbors

b. Node 0 receives "Clust_Reply" from all the neighbors

c. Node 0 sends "Clust_Comfirm" to cluster A and B. "Clust_Request" is flooded within A and B

d. Node 0 receives "Clust_Reply" from the rest of cluster A and B

e. Node 0 joins cluster A and sends "Clust_Update"

f. Node 4 send "Clust_Request" to cluster A and B

Figure 1: A simple example of SDC protocol (*TTL* = 2).

5.1 Experiment Settings

To test the applicability of our clustering protocol to different network structures, we use two types of topologies: Waxman topologies from GT-ITM topology generator [7] and power-law topologies from the BRITE generator [24].

We implement both the SDC and CDC algorithms and evaluate their performance on different types of topologies. There are several configurable parameters for the CDC scheme: *Vicinity, TwoHopThreshold, WeightThreshold* and *MinWeight*. These parameters affect the performance of CDC significantly: increasing the values of these parameters reduces the number of discovered clusters and causes more orphan nodes. We tune these parameters carefully towards the best clustering accuracy of CDC. Specifically, we set the values of *Vicinity, TwoHopThreshold, WeightThreshold, MinWeight* as: 1, 0.1, 0.0001, 0.0001 respectively. Besides these parameters, TTL is also critical to CDC as it affects the clustering accuracy by controlling he granularity of discovered clusters. Based on [27] and our observations, higher TTL values correspond to more accurate clusters with the tradeoff of increased number of messages. A TTL of 2 is used as the accuracy of CDC is not affected significantly by higher TTL values while the message overhead is much smaller than the overhead caused by higher TTL values.

We simulate the fully distributed systems in which a node only knows its directly connected neighbors. The one way 1-hop delay of any exchanged message is set to 1 simulated time.

The performance of SDC is evaluated in two network scenarios:

1. Static system where network nodes form a fixed topology throughout the whole simulation. In the beginning, every node shows up as an orphan node and starts its own clustering procedure independently at a random time $t \in [0, T]$. The simulation ends until every node is clustered and no message is exchanged.
2. Dynamic system in which the topology is changed by adding *X* new nodes to or removing *X* existing nodes from the system.

The main metrics used to evaluate the performance of the two algorithms are: *SCM*, *Message Overhead* and *Convergence Time*. SCM is used to evaluate the accuracy of the algorithm. Message Overhead is defined as the number of exchanged messages among nodes. Convergence time is the amount of simulated time from the first clustering operation until the end of the simulation. We also study the influence of node degree and TTL on the performance of SDC.

5.2 Performance of SDC in Static Systems

We first simulate the performance of SDC in clustering static systems, i.e., no node enters or exits the network. We are interested in the performance of SDC on different topology structures and topology sizes.

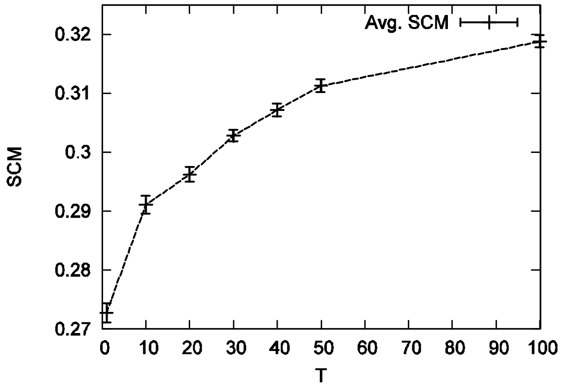
5.2.1 Affect of Clustering Time Span T

In SDC, a node may receive multiple clustering requests simultaneously especially when all the nodes start clustering in a short period of time, i.e. *T* is small. When receiving multiple requests, a node randomly picks one to serve and rejects the others. The smaller the value of *T*, the more rejected requests and control messages. Therefore, the value of *T* is a factor that can influence the performance of SDC significantly. We conduct simulations for different value of *T*, starting from 1 to 100 using a power law topology with 1000 nodes. The clustering accuracy, message overhead and convergence time are shown in the Fig. 2 □ 4.

As shown in Fig.2, increasing the value of *T* can improve SDC's clustering accuracy. The reason is straightforward. In the scenario of large *T*, a node can get service from more neighboring clusters due to the less competitions and thus is more likely to choose the best cluster to join. With the increase of *T*, the SCM keeps raising towards the maximum SCM value of the topology with a slower rate.

Fig.3 shows the message overhead under different value of *T*. When *T* is small (less than 10 for our simulation), the message overhead shows an increasing trend with the value of *T*. The reason can be explained as follows: When *T* is small, there are a lot of simultaneous clustering requests in the same area of the topology. Based on the SDC algorithm, only 1 request can be served at a time and the others have to be rejected and served later. Therefore, when *T* is small, a lot of clustering requests must be rejected which results in high control message overhead and many requests must be re-sent which also increases the message overhead. With the increase of *T*, the number of simultaneous requests in the same area is reduced, which contributes to the decreased message overhead. We can also observe a point in the value of *T* after which the message overhead shows an increasing trend. When *T* is large, although the overhead caused by simultaneous requests in the same area is reduced, the clustering operations are less aggregated. For example, a node n_i may postpone its moving attempt to a cluster C_j due to its neighbor's current clustering operation. If its neighbor also joins the cluster C_j, node n_i should perform another moving attempt that can be aggregated with the previous postponed moving operation. With the increase of *T*, clustering operations are less aggregated which increases the exchanged messages. When *T* is large enough, the increased number of messages due to the less clustering aggregation becomes more significant than the reduced number of messages due to the less simultaneous clustering requests and the overall message overhead shows an increasing trend.

The last metric we are interested in is the convergence time under different value of *T*. With the increase of *T*, the convergence time grows slowly. When *T* is larger than 30, the convergence time shows a rough linear correlation with the value of *T*. We can observe that the clustering procedure is more time consuming when *T* is small. The reason is small *T* causes simultaneous clustering attempt at more nodes, which further results in more time for handling the simultaneous requests and re-clustering.



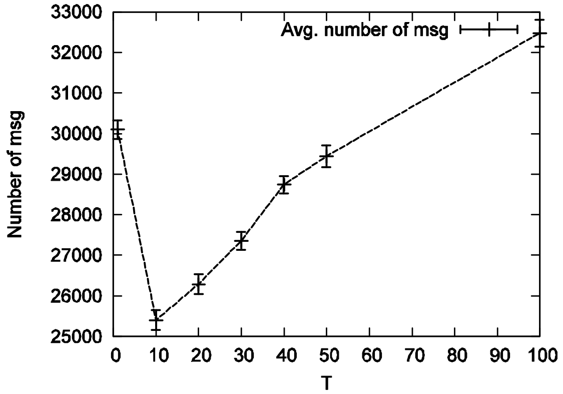


Figure 2: Clustering accuracy on power laws topology of 1000 nodes

Figure 3: Message overhead on power law topology of 1000 nodes

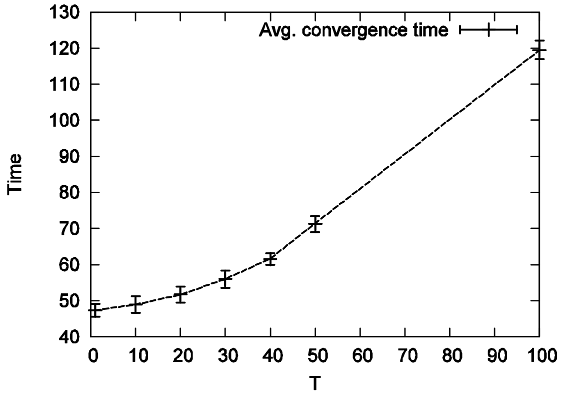


Figure 4: Convergence time on power law topology of 1000 nodes

5.2.2 Performance in Different Topology Size

In this set of simulations, we want to evaluate the performance of SDC in handling different size of topologies. Two types of topologies are used: random topologies from the Waxman topology model and power law topologies from the BA model. Both sets of topologies scale from 1000 nodes to 6000 nodes. We use the existing distributed clustering algorithm CDC as the reference point of our evaluation.

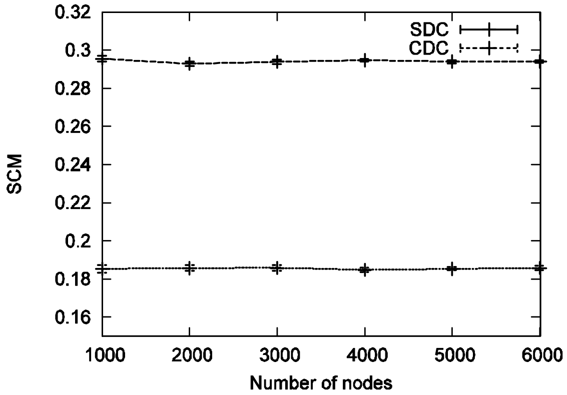
We first evaluate the performance of SDC on Waxman topologies. In this type of topologies, nodes are connected in a random way: only the distance information is considered. The average node degree is controlled to around 4 for all topologies.

The performance of both algorithms are shown in the Fig.5 □ 7. Since the topology structure is unchanged with the topology size, both algorithms have very stable clustering accuracy as shown in Fig.5. When compare the accuracy of the two algorithms, it is obvious that SDC performs a lot better than CDC. Recall that the range *T* for SDC is set to 2, so we expect even better performance of SDC in the scenario of larger *T*.

Fig.6 shows the message overhead of both algorithms. With the increase of topology size, the message overhead of SDC and CDC has a linear growing trend. Compared with CDC, SDC generates much lower message overhead which increases with topology size slowly. On the other hand, the message overhead of CDC is very high and increases rapidly with topology size as it is a flooding-based method.

The only metric in which CDC outperforms SDC is the convergence time. For CDC, clusters are formed right after message flooding. Therefore, the convergence time of CDC is determined only by the TTL of flooding and is unchanged with topology size. For SDC, since a node can serve only 1 clustering request at a time, the convergence time increases with the number of nodes. As multiple clustering requests at different areas of the topology can be served simultaneously, SDC's convergence time has a sublinear correlation with topology size.

Fig.8 □ 10 shows the performance of both algorithms on power law topologies. This type of topologies have very skewed degree distributions. For a BA topology with 5000 nodes, the highest degree is more than 100 while the average degree is only 4. The clustering accuracy of SDC on the power law topologies is slightly lower than on the Waxman topologies. The reason is the power law topologies have less significant clustering features that result in lower SCM values. Compared to CDC, the performance of SDC is consistently better in terms of clustering accuracy and message overhead. The convergence time is reduced on power law topologies but still higher than CDC.



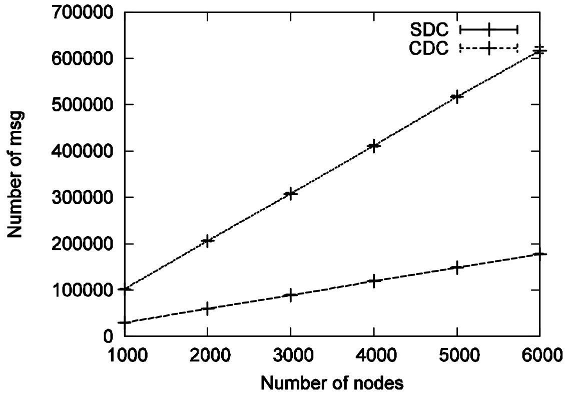


Figure 5: Clustering accuracy on Waxman topologies

Figure 6: Message overhead on Waxman topologies

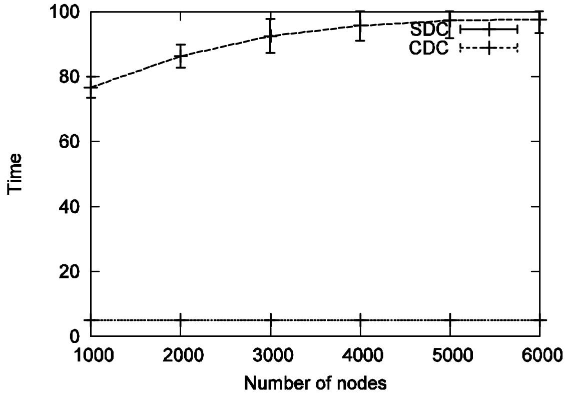
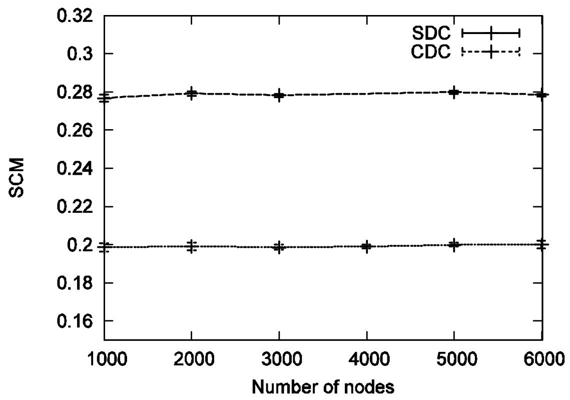


Figure 7: Convergence time on Waxman topologies



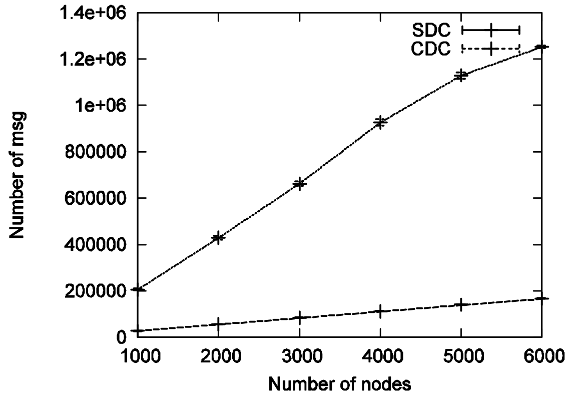


Figure 8: Clustering accuracy on BA topologies **Figure 9:** Message overhead on BA topologies

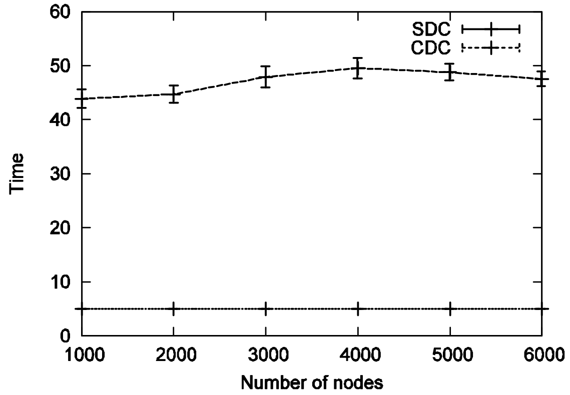


Figure 10: Convergence time on BA topologies

5.3 Performance of SDC in Dynamic Systems

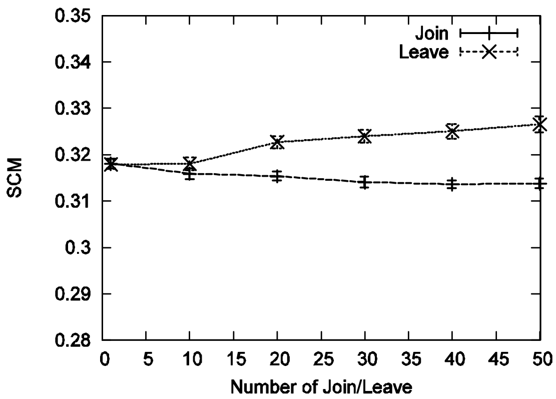
This set of simulations are conducted to evaluate the performance of SDC in handling node dynamics. The topologies used in this section are power law and Waxman topologies with 1000 nodes. To simulate the Join event, we first cluster the initial topology using the SDC algorithm and then we add *X* new nodes simultaneously to the network. Each of the new nodes connects with the existing nodes independently based on the topology model so that the topology structure can be maintained. For the Leave event, we randomly remove *X* existing nodes simultaneously from the topology. The simulations end when there is no message exchanged. We change the value of *X* from 1 to 50 and measure the clustering accuracy after node dynamics, message overhead and convergence time since the first Join/Leave event takes place.

Fig.11 □ 16 show the performance of SDC on handling node dynamics in a power law topology. The clustering accuracy after the simultaneous Join/Leave events is slightly changed, which indicates SDC is able to maintain accurate clustering from simultaneous node dynamics. The logic is that the topology structure is not changed significantly and therefore, accurate clustering must result in an SCM value that is close to the initial SCM before node dynamics. We also observe the SCM value after the Leave events increases slightly with the number of removed nodes. This performance is reasonable because removing a few existing edges makes the clustering features of the topology clearer. Since SDC can accurately form clusters, an increased SCM value that is consistent with the more significant clustering features can be observed. Following the same logic, when adding new nodes to the topology, the clustering features become less significant, which results in the slightly reduced SCM value.

Fig.12 shows the message overhead for node dynamics in the power law topology. For both Join and Leave events, the message overhead shows a linear correlation with the number of dynamic nodes. Moreover, the Leave events cause more exchanged messages than the Join events. This is because more nodes are affected and need to re-cluster after a Leave event than after a Join event. After a node leaves, all of its previous neighbors and all the nodes in its original cluster should re-cluster but after a new node joins the topology, only its neighbors are affected and re-cluster.

Fig.13 shows the convergence time of SDC for node dynamics. With the increase of Join/Leave events, the convergence time increases slowly. We can also observe that Leave events take longer to converge than Join events because more nodes are affected by a Leave event and need to re-cluster.

The performance on the Waxman topology is consistent with the performance on the BA topology. We can see the advantage of SDC in handling dynamic systems: With low message overhead, accurate clusters can be maintained after a different number of simultaneous Join and Leave events. This performance is especially suitable for a system with continuous node entry and exit. To maintain an acceptable clustering accuracy, the existing algorithm CDC must re-cluster the whole network after a certain number of node entry and exit, which causes a high message overhead and is not scalable for large networks.



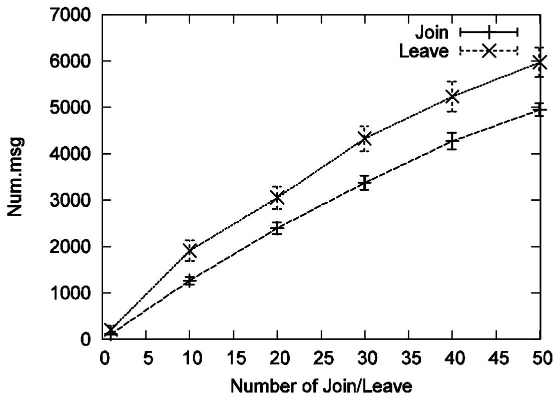


Figure 11: Clustering accuracy for node dynamics on BA topology with 1000 nodes

Figure 12: Message overhead for node dynamics on BA topology with 1000 nodes

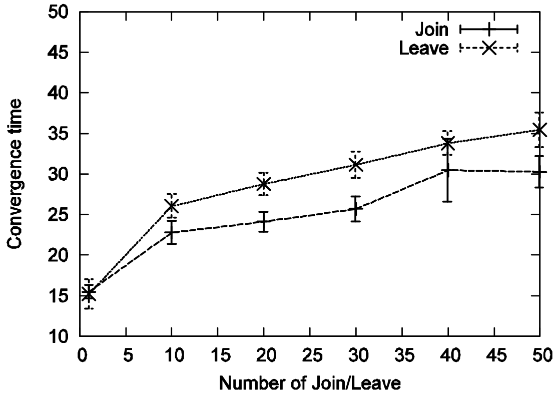
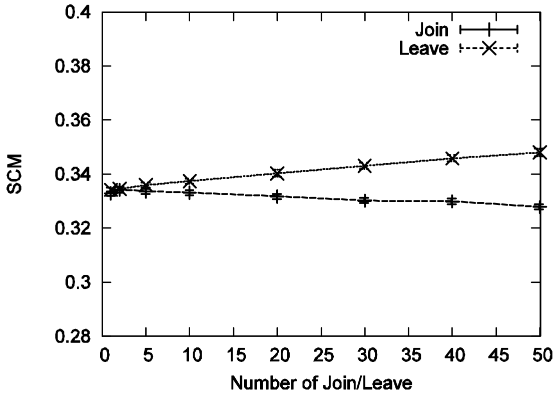


Figure 13: Convergence time for node dynamics on BA topology with 1000 nodes



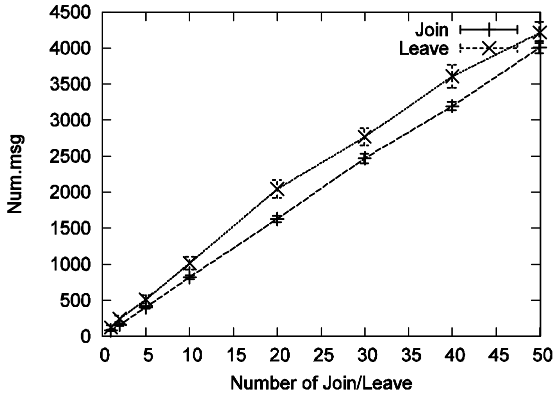


Figure 14: Clustering accuracy for node dynamics on Waxman topology with 1000 nodes

Figure 15: Message overhead for node dynamics on Waxman topology with 1000 nodes

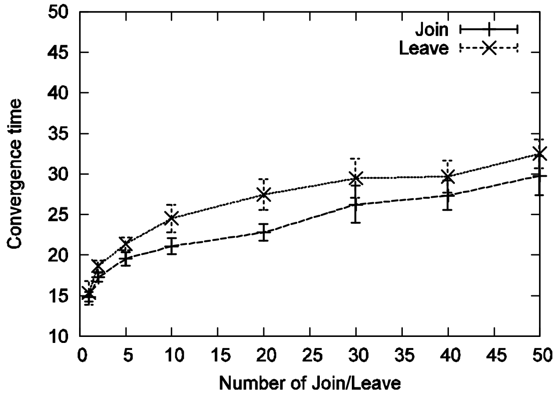
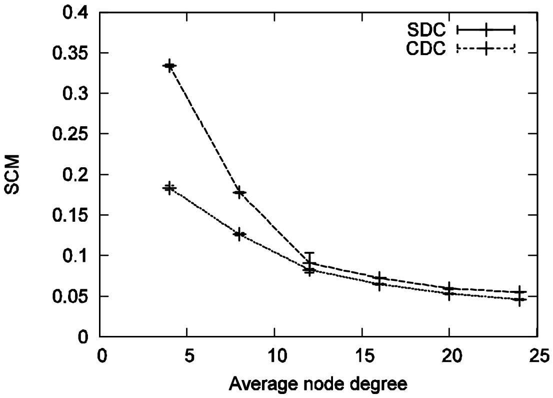


Figure 16: Convergence time for node dynamics on Waxman topology with 1000 nodes



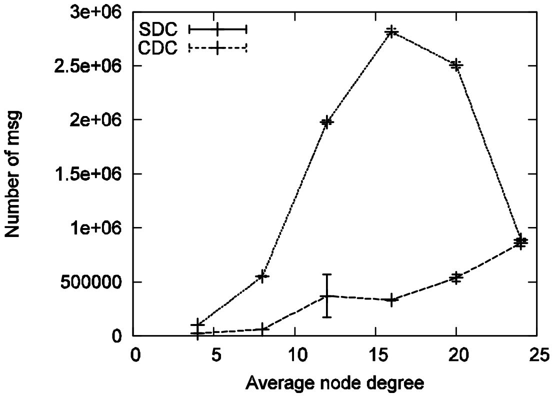


Figure 17: Effect of node degree on clustering accuracy

Figure 18: Effect of node degree on message overhead

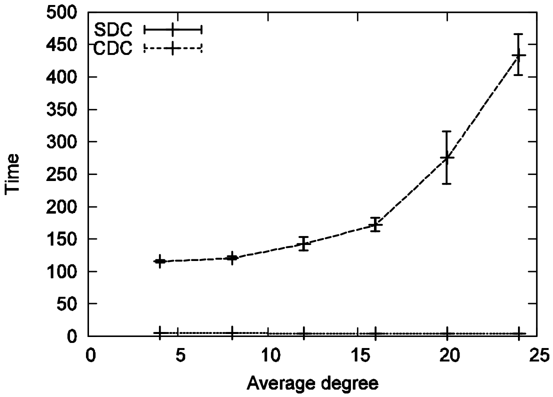


Figure 19: Effect of node degree on convergence time

5.4 Influence of Node Degree

Node degree is an important factor to the performance of SDC since the clustering procedure is based on node connectivity. In this set of simulations, we study how different average node degree can affect the performance of the algorithm.

The topologies used in this section are Waxman topologies with 1000 nodes and different average degree ranging from 4 to 24. The range *T* is set to 100. Again, we use CDC as the reference point of the evaluation.

Fig. 17 shows the clustering accuracy of the two algorithms against different average node degrees. Clearly, the SCM values of both SDC and CDC drop with the increase of average degree. This decline of SCM is mainly caused by the decrease in clustering features of the topology other than the clustering algorithms. Since a connection is determined in a random manner by the Waxman model, increasing the average degree adds more randomness to the topology as translated to less clustering features. However, SDC can capture the clustering features better than CDC as shown by the higher SCM values, especially when the clustering features are more significant.

Fig.18 shows the message overhead of both algorithms when clustering topologies with different average degrees. It is shown that the increase in average degree leads to higher message overhead for SDC. This fact is under our expectation. In SDC, after a node finishes its current clustering operation, its neighbors need to start a new round of clustering procedures. If a topology has higher average degree, more nodes are involved in each clustering procedure and need to take actions after the current clustering, which leads to more message overhead. When compare the two algorithms, we claim a better performance of SDC as the generated messages in SDC are much less than in CDC. A quick drop in the message overhead of CDC can be observed when the average degree exceeds 16, which is caused by the parameter setting. The convergence time of SDC also inclines with the increase of average node degree due to the same reason.

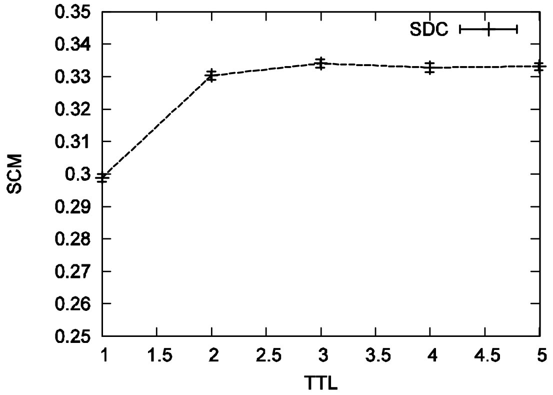
As a summary, SDC is able to detect accurate clusters in both sparse and dense topologies. The message overhead and convergence time do increase with average node degree due to increased number of nodes involved by each clustering operation.

5.5 Influence of TTL

In SDC, a node's clustering request is rejected if the TTL of the request message expires. Therefore, clustering results are affected by the value of TTL. Intuitively, large TTL values do not influence clustering accuracy because nodes can always join a cluster that leads to the highest SCM value without being rejected due to the expire of TTL. The questions studied in this section are how large the TTL should be to guarantee accurate clustering and how the other performance metrics are affected by different TTL values. We cluster a 1000 node Waxman topology using SDC with TTL varied from 1 to 5 and measure the performance metrics.

Fig.20 shows the clustering accuracy of SDC against different TTL. It is under our expectation that the SCM is improved with the increase of TTL. The most significant SCM improvement happens when TTL changes from 1 to 2. Obviously, for the Waxman topology in our simulation, clusters with 1-hop diameter are not accurate at all. When we further increase TTL, the SCM value does not benefit from higher TTLs. Based on the definition of SCM, accurate clusters do not have large diameters because the number of non-neighbor nodes should be minimized. Thus, clusters do not grow further when their diameter reaches 2, which leads to the stable SCM values at TTL of 2 and higher.

When we examine the message overhead, we observe positive effect of increasing TTL as shown in Fig.21. The steady decrease of message overhead is a result of reduced $Clust\_Reject$ messages due to higher TTL values. Similar to SCM, the most significant reduction happens when TTL increases from 1 to 2 and the message overhead stabilizes when TTL is further increased. These results can be explained as follows: When TTL is 1, many clustering requests are rejected due to TTL expiration, which results in high message overhead. When we increase TTL from 1 to 2, the $Clust\_Reject$ messages are reduced significantly and the clusters are able to grow towards the highest accuracy, which results in the significant drop in message overhead. Further increase in TTL has little effect on message overhead because most clusters stop growing when diameter reaches 2 and TTL no longer expires. The convergence time under different TTL values is shown in Fig.22 that can be explained in the same way as we do for message overhead.



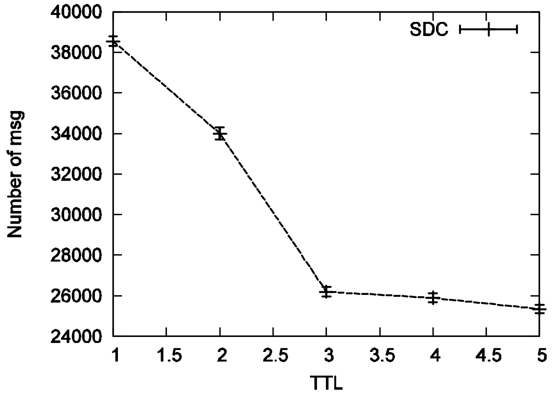


Figure 20: Effect of TTL on clustering accuracy **Figure 21:** Effect of TTL on message overhead

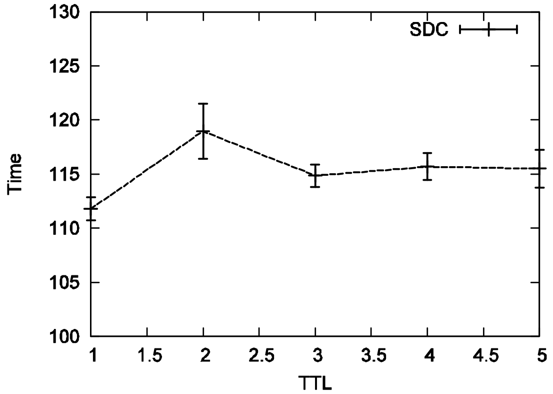


Figure 22: Effect of TTL on convergence time

6. CONCLUSIONS

In this paper, we target the challenging problem of clustering large-scale distributed systems such as P2P networks. We identify the main issues in the existing clustering algorithms: First, many existing algorithms are centralized methods and therefore they are not scalable and efficient in handling large-scale distributed systems. Second, although several distributed clustering algorithms have been proposed, they can not guarantee accurate clustering and low message overhead especially when handling dynamic systems. Therefore, we propose a novel distributed clustering protocol SDC for large-scale distributed systems. We conduct extensive simulations to evaluate the performance of SDC under various scenarios. The simulation results show the promising performance of SDC: it is a scalable and accurate clustering approach with small message overhead on various topologies. The most attractive feature of SDC is that it can reserve high clustering accuracy from multiple simultaneous node entry and exit with small message overhead.

7. REFERENCES

[1] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30:2826–2841, June 2007.

[2] J. Ahuja and J.-H. Cui. A scalable peer-to-peer file sharing system supporting complex queries. *UCONN CSE Technical Report*, *UbiNet TR05-01*, January 2005.

[3] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, December 2004.

[4] R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance of complex networks. *Discrete Applied Mathematics*, 406:378–382, August 2000.

[5] A. Barbu and S.-C. Zhu. Graph partition by swendsen-wang cuts. *Ninth IEEE International Conference on Computer Vision Volume 1*, 10 13 - 10 2003, Nice, France.

[6] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85(25):5468–5471, Dec 2000.

[7] K. Calvert and E. Zegura. Gt-itm: Georgia tech internetwork topology models. *http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz*, 1996.

[8] M. Chatterjee, S. K. Das, and D. Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5:193–204, April 2002.

[9] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, 2002.

[10] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Inf. Process. Lett.*, 61(3):121–127, 1997.

[11] N. Dimokas, D. Katsaros, and Y. Manolopoulos. Node clustering in wireless sensor networks by considering structural characteristics of the network graph. *International Conference on Information Technology*, 00:122–127, 2007.

[12] D. Doleva, S. Jaminb, O. Mokrync, and Y. Shavittc. Internet resiliency to attacks and failures under bgp policy routing. *Computer Networks*, 50:3183–3196, November 2005.

[13] D. Estrin, Y. Rekhter, and S. Hotz. Scalable inter-domain routing architecture. *In SIGCOMM '92: Conference proceedings on Communications architectures & protocols*, pages 40–52, 1992.

[14] Y. Fernandess and D. Malkhi. K-clustering in wireless ad hoc networks. *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, 2002.

[15] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. *In IEEE INFOCOM*, volume 2, pages 1455–1466, March 2005.

[16] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller. Hierarchical p2p systems. *In Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, 2003.

[17] C. Gkantsidis, M. Mihail, , and E. Zegura. Spectral analysis of internet topologies. *IEEE INFOCOM,* 2003.

[18] R. Hoes, T. Basten, W.-L. Yeow, C.-K. Tham, M. Geilen, and H. Corporaal. Qos management for wireless sensor networks with a mobile sink. *In EWSN '09: Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 53–68, Berlin, Heidelberg, 2009. Springer-Verlag.

[19] J. Jin, J. Liang, J. Jin, and K. Nahrstedt. Large-scale qos-aware service-oriented networking with a clustering-based approach. *In 16th International Conference on Computer Communications and Networks*, pages 522–528, August 2007.

[20] V. Kantere, D. Tsoumakos, T. Sellis, and N. Roussopoulos. Groupeer: Dynamic clustering of p2p databases. *Information Systems*, 34:62–86, March 2009.

[21] G. Kwon and K. D. Ryu. An efficient peer-to-peer file sharing exploiting hierarchy and asymmetry. *In SAINT*, pages 226–233, 2003.

[22] Y. Li, J.-H. Cui, D. Maggiorini, and M. Faloutsos. Characterizing and modeling clustering features in as-level internet topology. *In Proceedings of IEEE INFOCOM*, 2008.

[23] A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communication*, Vol. 17, No. 8, August 1999.

[24] A. Medina, A. Lakhina, I. Matta, , and J. Byers. Brite: Universal topology generation from a user's perspective. *In Proceedings of Workshop the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, October 2001.

[25] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. *In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM.

[26] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86(14):3200–3203, Apr 2001.

[27] L. Ramaswamy, B. Gedik, and L. Liu. A distributed approach to node clustering in decentralized peerto-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(9), Sept. 2005.

[28] S. van Dongen. A new cluster algorithm for graphs. *Technical report INS-R9814, Centrum voor Wiskunde en Informatica (CWI)*, ISSN 1386-3681, Dec. 1998.

[29] S. van Dongen. Performancde criteria for graph clustering and markov cluster experiments. *Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam*, 2000.

[30] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. *In IEEE INFOCOM*, 2004.

[31] W. Zheng, S. Zhang, Y. Ouyang, F. Makedon, and J. Ford. Node clustering based on link delay in p2p networks. *In SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 744–749, 2005.

[32] C. C. Zou, D. Towsley, and W. Gong. Modeling and simulation study of the propagation and defense of internet email worm. *IEEE Transactions on Dependable and Secure Computing*, 4:105–118, 2007.

Replica Placement In Unstable Radio Links

K.V.Ramana M.Tech vamsivihar@gmail.com
Associate Professor/ CSE Department
Jawaharlal Nehru Technological University
Kakinada, 533003,India.

Dr. Raghu B.Korrapati Ph.D raghu.korrapati@gmail.com
Walden University

S.Hemanth M.Tech hemanth37@jntuk.edu.in
Jawaharlal Nehru Technological University
Kakinada, 533003,India.

N. Praveen Kumar praveen2218@jntuk.edu.in
Jawaharlal Nehru Technological University
Kakinada, 533003,India.

Abstract

The growth in wireless communication technologies has immensely gained a lot of attention in ad-hoc networks especially in the area of mobile hosts. As the network topology in this type of ad-hoc networks changes dynamically, chances of network getting disconnected is also very frequent due to unstable radio links for shorter time intervals. In this paper, we proposed a succinct solution for the problem replica allocation in a mobile ad-hoc network by exploring group mobility. The solution for this problem is carried out in three phases. Thus, with this solution even if the hosts get disconnected we can replicate the data items on to mobile hosts so that the mobile hosts can still access the data. Several experiments are conducted to evaluate the performance of the proposed scheme. The experimental results show that the proposed scheme is able to not only obtain higher data accessibility but also produce lower network traffic than prior schemes.

Keywords: Ad hoc network, Replication, Radio links, Transfer cost

1. INTRODUCTION

An ad hoc network is an autonomous system of mobile nodes/routers (and associated hosts) connected [1] by wireless links. Ad hoc or short live network is a collection [2] of autonomous wireless mobile hosts or routers communicating with each other dynamically and freely forming a multi-hop, fully self-organized, temporary meshed network with no human intervention and with no existing pre-established fixed communication infrastructure. The nodes in ad hoc wireless networks are free to move [3] independently in any direction. The network topology may change randomly at unpredictable times and primarily bidirectional links. These networks have lower bandwidth capacity and shorter transmission [4] range than fixed infrastructure networks. The throughput of wireless communication is lesser than wired communication because of the effect of the multiple access, fading, noise, and interference conditions.

In an ad hoc network every mobile host acts as a router and maintains [5] its neighbor hosts in a routing table. If one host enters into another hosts range then the routing tables of those hosts are updated with each other details. If one of the hosts in the networks, expect data from the another host which is in the same network, there is no need to communicate directly with their radio range, data is still forwarded to the destination mobile host by relaying transmission through other mobile hosts which exist between the two mobile hosts.

Mobile hosts are continuously moving [6] from one place to another place in the same network, and then there exist frequent disconnections among the mobile hosts due to unstable radio links. Some connections get lost and some new connections are establishing among mobile hosts. Due to these frequent disconnections by unstable radio links, it's difficult to access the data from the source node. Since one cannot control network disconnections, an alternative solution to this problem is to replicate data onto mobile hosts so that when disconnections by unstable radio links occur, mobile hosts can still access data.

Several reports have been proposed on caching [7] data onto mobile hosts. In these reports a network has been assumed in which mobile hosts are frequently moved and get disconnected with neighbor mobile hosts due to unstable radio links.

2. RELATED WORK

Previously different strategies for caching data contents in mobile computing environments have also been proposed [8,9,10,15,16,17]. Most of these strategies assume an environment where mobile hosts accesses contents at sites in a fixed network, and cache data on the mobile hosts because wireless communication is more costly than wired communication. Such strategies address the issue of keeping consistency between original data and its replicas or caches with low communication costs. They are considered to be similar to our approach, because both approaches replicate data on mobile hosts. However, these strategies assume only one-hop wireless communication, and thus, they are completely different from our approach that assumes multi hop ad hoc communication.

Another closely related research topic is of push-based information systems in which a server repeatedly broadcasts data to clients using a broadband channel. Several caching strategies have been proposed to improve user access [11, 12]. In these strategies, clients are typically mobile hosts, and the cache replacement is determined based on several parameters such as the access frequency from each mobile host to each data item, the broadcast frequency of each data item, and the time remaining until each item is broadcast next. However, comparing the strategies for caching or replicating, both approaches are completely different because the strategies in push-based information systems do not assume that the clients cooperatively share cached data in ad hoc networks.

In another research topic, a mathematical model for data object replication in ad hoc networks is formulated. This model, proposed a game theoretical technique in which players (mobile hosts) continuously compete in a non-cooperative environment to improve data accessibility by replicating data objects. This is very closely related work to our mechanism, because in this research, an ad hoc network has been considered, where mobile hosts are moving continuously, network suffers from frequent disconnections. For this problem a solution has been found by replicating data items on neighbor mobile hosts. In general the mobile hosts would experience reduced access latencies provided that data is replicated within their close proximity. However, this is applicable in cases when only read accesses are considered. If updates of the contents are also under focus, then the locations of the replicas have to be 1) in close proximity to the mobile hosts, and 2) in close proximity to the primary (assuming a "master" replication environment [13]) copy. Therefore, efficient and effective replication schemas strongly depend on how many replicas to be placed in the system, and more importantly where [14]. This topic is different from our approach, because we consider an ad hoc network where mobile hosts are moving continuously and suffers from short time disconnections due to unstable radio links.

3. METHODOLOGY

3.1 Modules

The proposed work has been segmented in to three phases. The first phase describes how the hosts are entered into the network and how an adjacent matrix is computed. The second phase establishes the communication between hosts. The third phase will explain about reading and updating access of data items among mobile hosts in an Ad Hoc network where hosts are continuously moving within the network and disconnections occur due to unstable radio links which are likely to be disconnected after a short time.

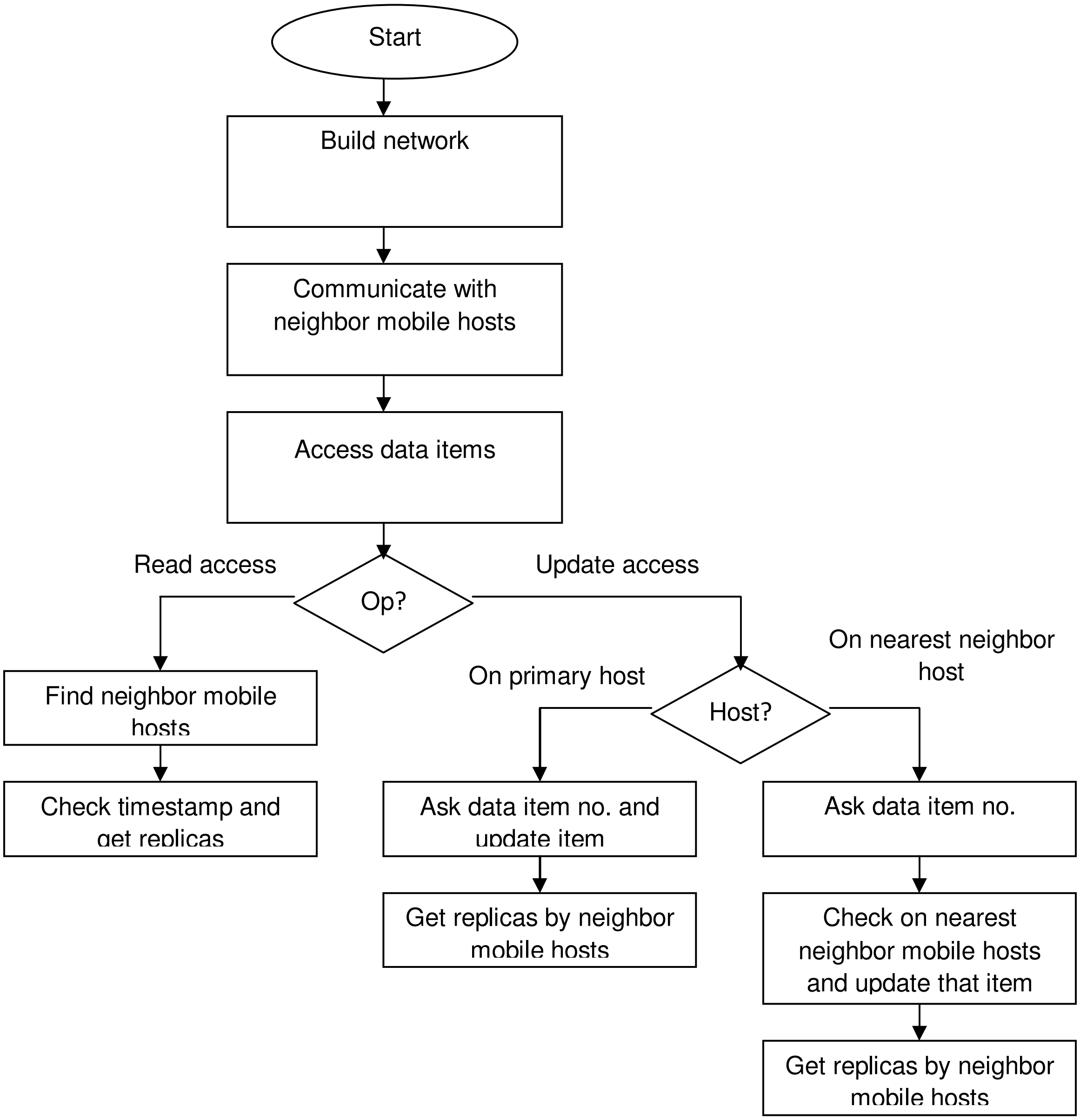

Start
Build network
Communicate with neighbor mobile hosts
Access data items
Read access
Op?
Update access
Find neighbor mobile hosts
Check timestamp and get replicas
On primary host
Host?
On nearest neighbor host
Ask data item no. and update item
Get replicas by neighbor mobile hosts
Ask data item no.
Check on nearest neighbor mobile hosts and update that item
Get replicas by neighbor mobile hosts

FIGURE 1: Flow chart to access data items.

Figure 1 represents the flow chart for performing operations on data objects through modules namely Build Network, Communication, Reading and Updating Data items.

The functionality of the proposed modules is described in below sections.

3.1.1 Build network

In the first phase a system is proposed by considering an ad hoc network where a new host enters into the network with its respective radio range. Let 'r' be the transmission range or radio range of the mobile node. By the host's radio range(r), one host is connected with another host which is in the host range and those are said to be neighbors to each others. Every host maintain two lists, one list is for data items, to which the host is primary host and second list for replicas, which data items have replicas on the host. The storage capacity of the host is greater than the total size of the replicas maintained on the mobile host. We maintain a matrix in which 1 represents neighbor mobile hosts to each other and 0 represents those hosts are not in their radio range(r). Using this matrix we can know which mobile hosts are neighbors to which mobile host. The matrix is termed as adjacent matrix where rows and columns represent mobile hosts in the Ad hoc network which is considered in the proposed system. The format of the adjacency matrix is represented in **FIGURE 6**.

3.1.2 Communication

In the second phase, a communication between the mobile hosts has been established. As considered network is ad hoc network the mobile hosts tend to move dynamically by its nature. Due to this a short time disconnections occur in the network then existing connections may be lost and new connections may be established or no other connections are established. Every time when mobile host moved from one place to another it checks the neighbor mobile hosts in its radio range(r), if there are same hosts exist in the radio range(r) then establish a connection and read data items from the neighbor mobile hosts. It also checks if it has already replica of that data items, then check the data item whether it is modified or not, if it was modified, then update that data items replica in its replica list.

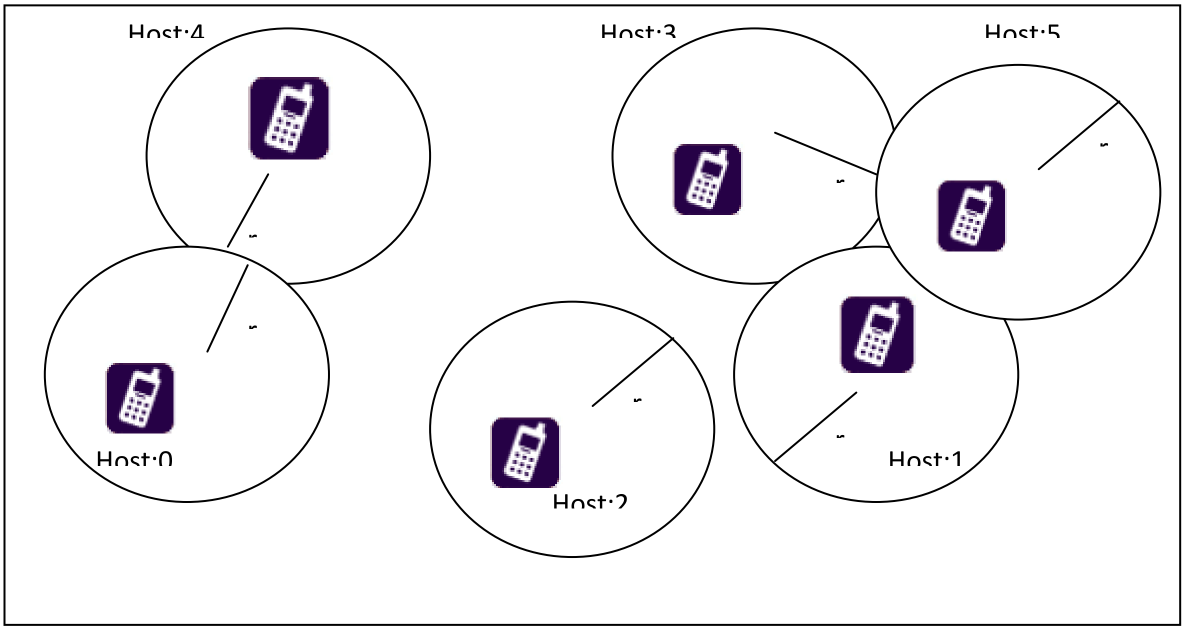


FIGURE 2: Communication of hosts in a Mobile Ad hoc network.

In figure 2 hosts 1,3,5 are neighbor mobile hosts which communicate each other and can exchange replicas, same as in hosts 4,6 replica exchange can take place, but host 2 was been isolated.

3.1.3 Reading and Updating Data items

In the third phase, reading and updating data items are considered. While transmitting the data items in non-replication systems, the data items temporarily stored in intermediate mobile host's buffer and forwarded to receiver's host. As most of the radio links are unstable, disconnections may occur and leads to loss of data.

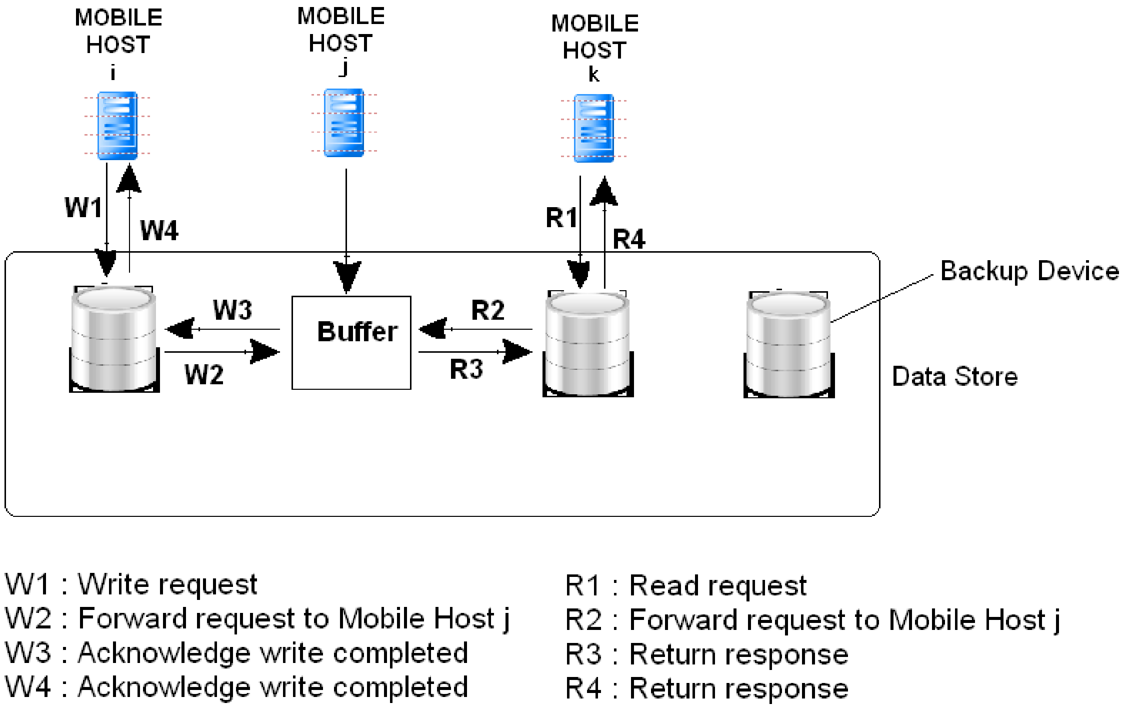


FIGURE 3: System that do not support Replication.

Figure 3 represents the phenomena of communicating data items from mobile host (i) to the mobile host (k) through the mobile host (j), where host(j) acts as a server host.

To rectify this problem, a mechanism, which stores the replica of data item in the intermediate mobile hosts. When reading data items as shown in figure 4, initially the client's checks its replica list itself. If replica item in the server mobile host is not presented in the client mobile host then the client host reads the replica of the data item from the server mobile host and adds this data item replica to the replica list maintained in the client mobile host.

If the replica of one data item is available in the server mobile host then client mobile host compares with the timestamp of the replicated data item in the server mobile host. If the timestamp of the replicated item in the server mobile host is previous than that of the client mobile host then client ignores the read access, otherwise client modifies replica item in the replicas list of client mobile host by reading the data item replica from the server mobile host.

Updating is considered, when the client mobile host(k) is neighbor to server mobile host(j) which must be a primary copy of the data item, which is to be updated. Then in this case, the server mobile host asks for new data to update.

Every time, when the mobile host moved from one place to another, it first checks the data items replicated on the neighbor hosts and then reads data items from the neighbor hosts. If update takes place on nearest neighbor then it performs three functions. First, it asks for data item number to be modified and verify which mobile host has that data item number's primary copy. Second, update that data item in the master host of that item and set the new time stamp to that data item and then send replica of the newly updated data item to the neighbor hosts. No other nearest neighbor has the data item number's primary copy then it can't be updatable.

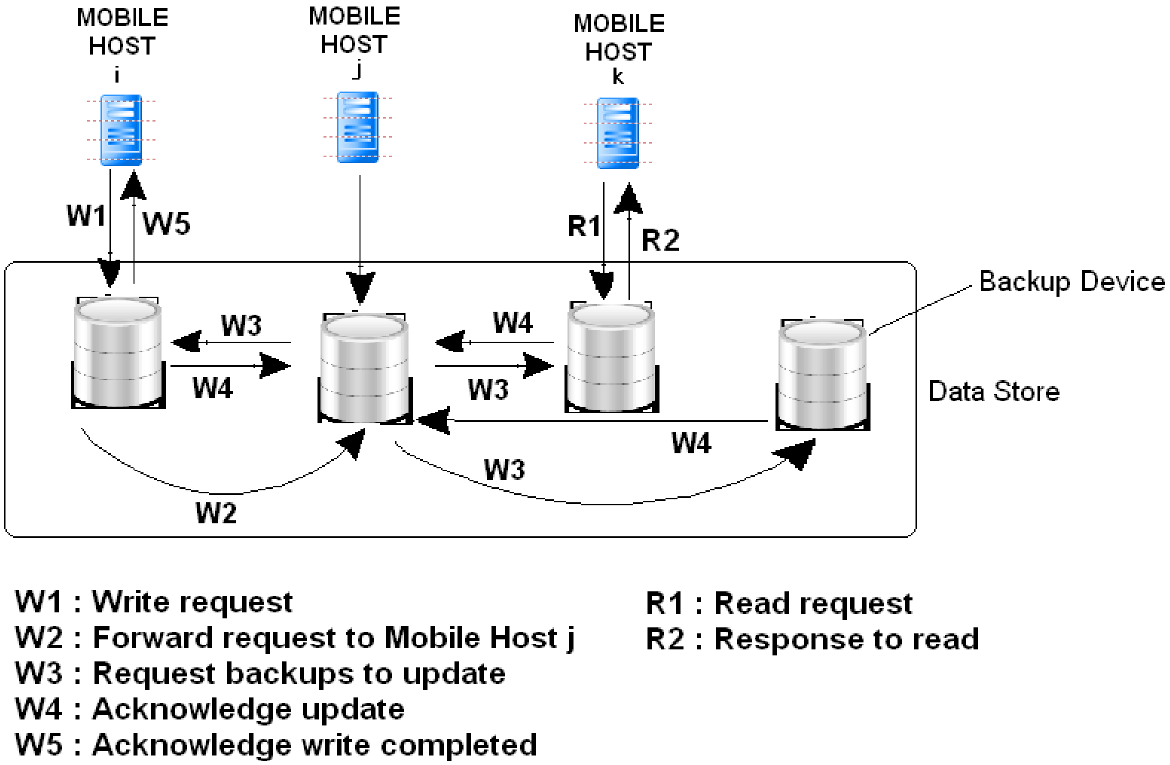


FIGURE 4: System that support Replication.

Figure 4 represents the same phenomena as of Figure 3 with additional request and response backup service to update the data.

Proposed Algorithm for Updating Data Items:

```
Initialization:
ItemsList,ReplicaList;
Curitem=null;itemidx=0;repidx=0;itemno=null;
UPDATEITEM()
get(item_no);
//update on primary host
    For each itemidx<items.size do
        Curitem=items.get(itemidex);
        If(itemno==curitem.itemno)
            get(newdata_to_update);
            set(newdata);
            set(timestamp);
        End if
        break;
    end for
//update on closest neighbour host
    For each repidex<replica.size do
        Curitem=replica.get(repidx)
        If(itemno==curitem.itemno)
```

```
            If(validNbr(curitem.primaryhost)
                get(newdata_to_update);
                set(newdata);
                set(timestamp);
        end if
        break;
    end for
```

Description of Algorithm

In this algorithm a data item can be updated only at two locations. The two locations can be update on primary host and update on nearest neighbor. On every mobile host in the network two lists are maintained. The first list consists of list of data items that have their size less than the total available storage on that mobile host. The second list consists of replicas list those are replicated on the mobile. An itemno is provided as input to update the data item and then its checks the data items list. If the item number is presented in the list, it is asks for new data and update the item number with new data and it gives a new timestamp to the updated data item. As shown in the algorithm.

To update on nearest neighbor a check is performed on the replica items list. If the item number is in the replica list then it checks for the data item primary host whether it is neighbor or not. If not that gives error message, otherwise it asks for new data item to update. The new data item updated, gives timestamp to the updated data item. After the completing update access, neighbor hosts gets replica of the updated data item.

Monitoring cost

Considering an Ad Hoc network in which m mobile hosts are moving continuously and disconnections occur due to unstable radio links. Every mobile host has its own storage capacity. When two mobile hosts communicate with each other then they give a positive number associates a communication cost c(i,j) where i, j are mobile hosts id's. when the upstream and downstream bandwidths are equal then c(i,j)=c(j,i). Let there be n data objects, and r and u ate the read and update accesses respectively. When a mobile host M^i initiates a read request for a data object, the request is redirected to the nearest neighbor NN_k^i mobile host that holds either the original or the replica of the data object. $NN_k^i= M^i$ if and only if M^i is a replicator or the primary mobile host of the data item.

We are interested in minimizing the total Network Transfer Cost (NTC) due to data object read and update access.

Let R_k^i represents the total NTC, due to M^is' reading request for data object O_k, addressed to the nearest neighbor host NN_k^i.

This cost is given by the following equation.

$$R_k^i = r_k^i o_k c(i, NN_k^i) \,, \qquad (1)$$

The second component of NTC is the cost arising due to the updates. Let U_k^i be the total NTC, due to P_ks' updates requests for object O_k.

$$U_k^i = u_k^i o_k \sum_{\forall i \in R_k} c(P_k, i) \,. \qquad (2)$$

The overall cost is denoted as below

$$C_{overall} = \sum_{i=1}^{m} \sum_{k=1}^{n} (R_k^i + U_k^i) . \qquad (3)$$

4. RESULTS AND DISCUSSIONS

In an ad hoc network the hosts continuously move with build dynamic topologies. Figure 5 shows the network model at a particular time instance of how seven hosts are connected.

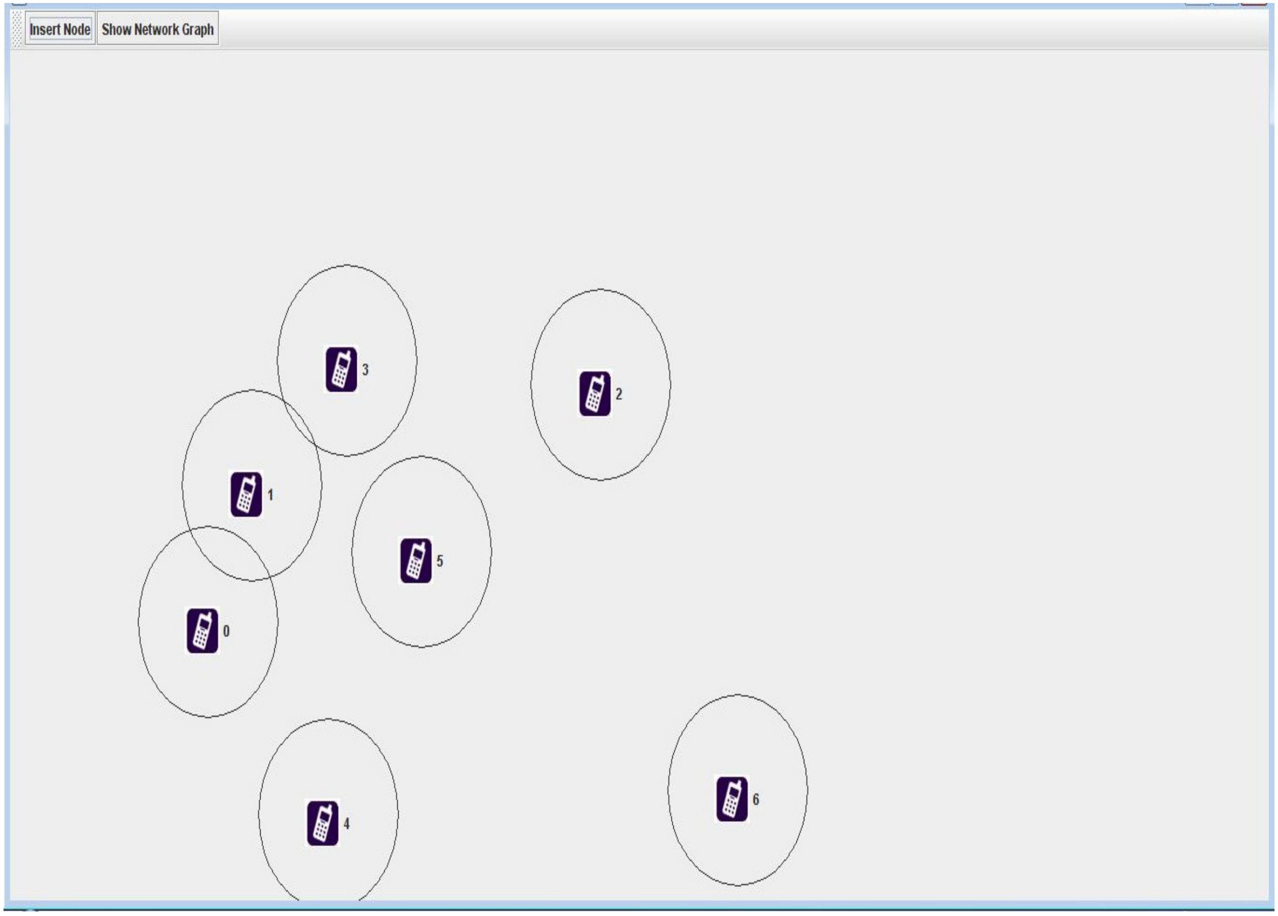
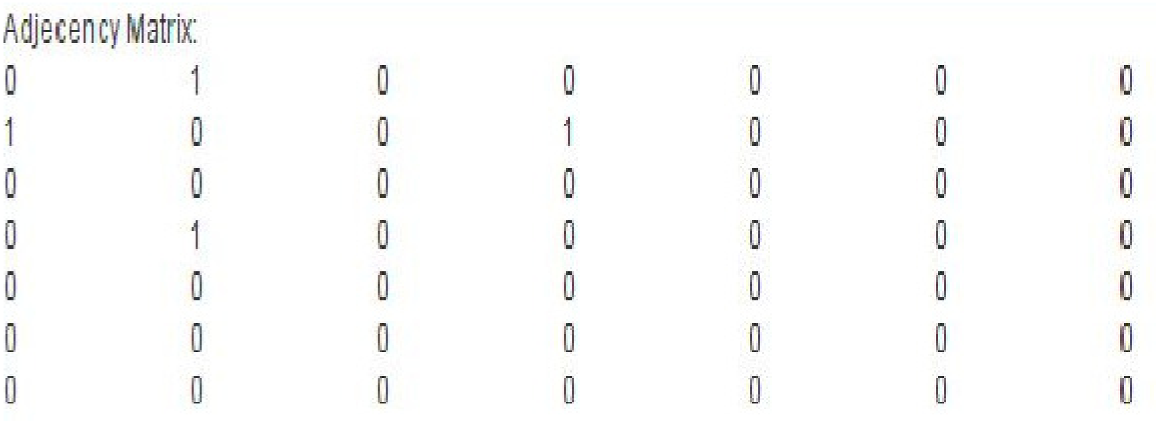


FIGURE 5: Sample Ad hoc network with seven hosts.

For the Figure 5 the adjacency matrix at a particular time instant is shown in Figure 6.

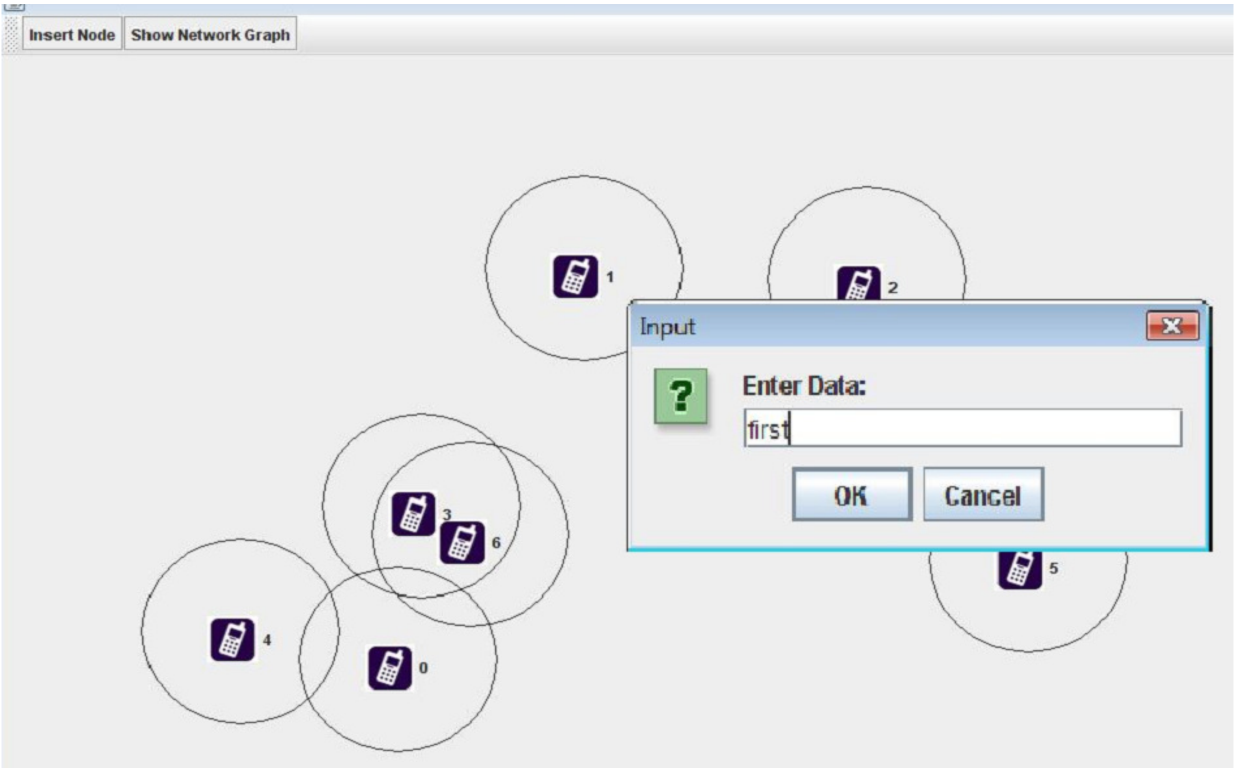


Adjecency Matrix:

0	1	0	0	0	0	0
1	0	0	1	0	0	0
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

FIGURE 6: Adjacency matrix for the sample ad hoc network.

Adjacency matrix in the Figure 6 represents which are neighbor to each other, ‘1’ represents connection between the hosts and ‘0’ represents

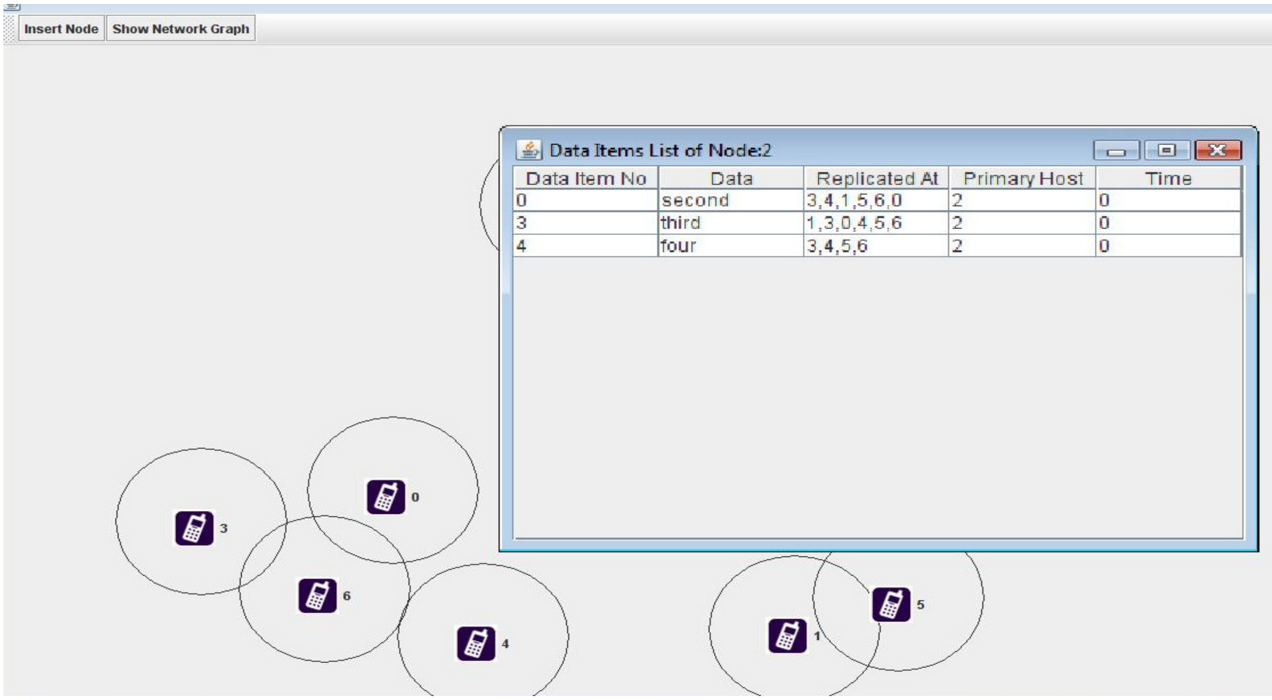


Input

Enter Data: first

FIGURE 7: Entering data on the host.

Figure 7 show the data item to enter, when data item entered it added to the data items list in the corresponding host and neighbor hosts get replicas from that host.

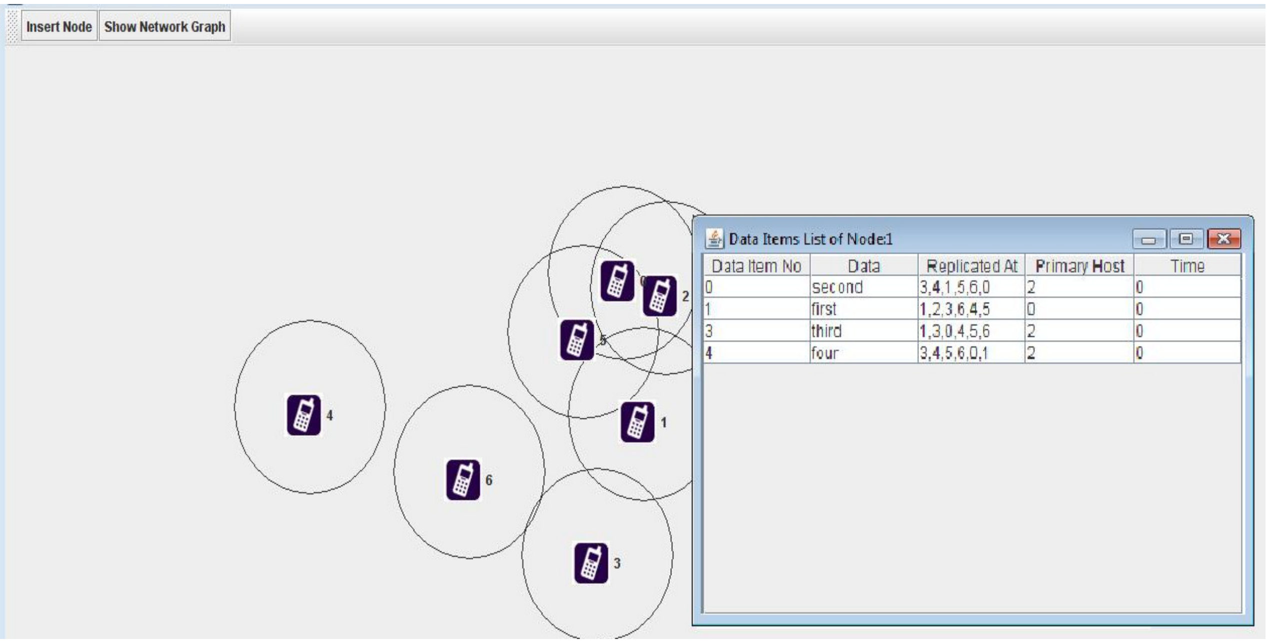


Data Items List of Node:2

Data Item No	Data	Replicated At	Primary Host	Time
0	second	3,4,1,5,6,0	2	0
3	third	1,3,0,4,5,6	2	0
4	four	3,4,5,6	2	0

FIGURE 8: Primary data items on host 2.

Figure 8 shows the primary data items on host 2 and the replica of primary data items of host 2 are on other hosts are shown in Figure 9.



Data Items List of Node:1

Data Item No	Data	Replicated At	Primary Host	Time
0	second	3,4,1,5,6,0	2	0
1	first	1,2,3,6,4,5	0	0
3	third	1,3,0,4,5,6	2	0
4	four	3,4,5,6,0,1	2	0

FIGURE 9: Replicas of data items.

Replicas of data items of host 2 on host 1 is shown in Figure 9

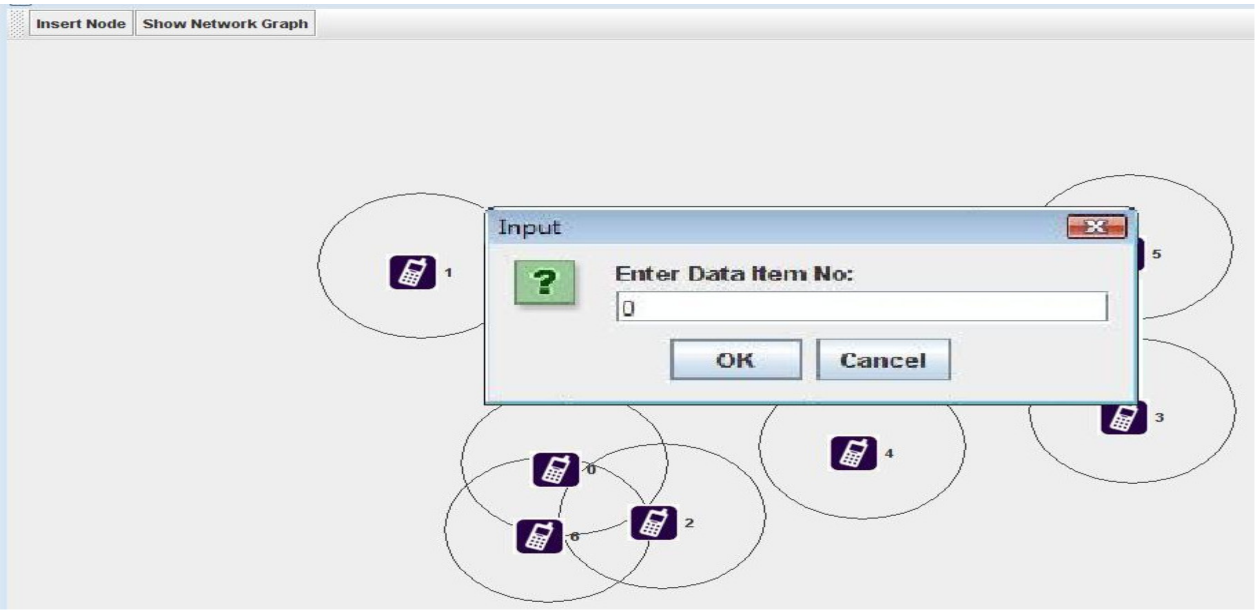


FIGURE 10: Enter data item number to update

Figure 10 explains the way the data item number entered to be updated on a particular host. In background the data item number verified that whether the entered data item number exists in the corresponding host data items list.

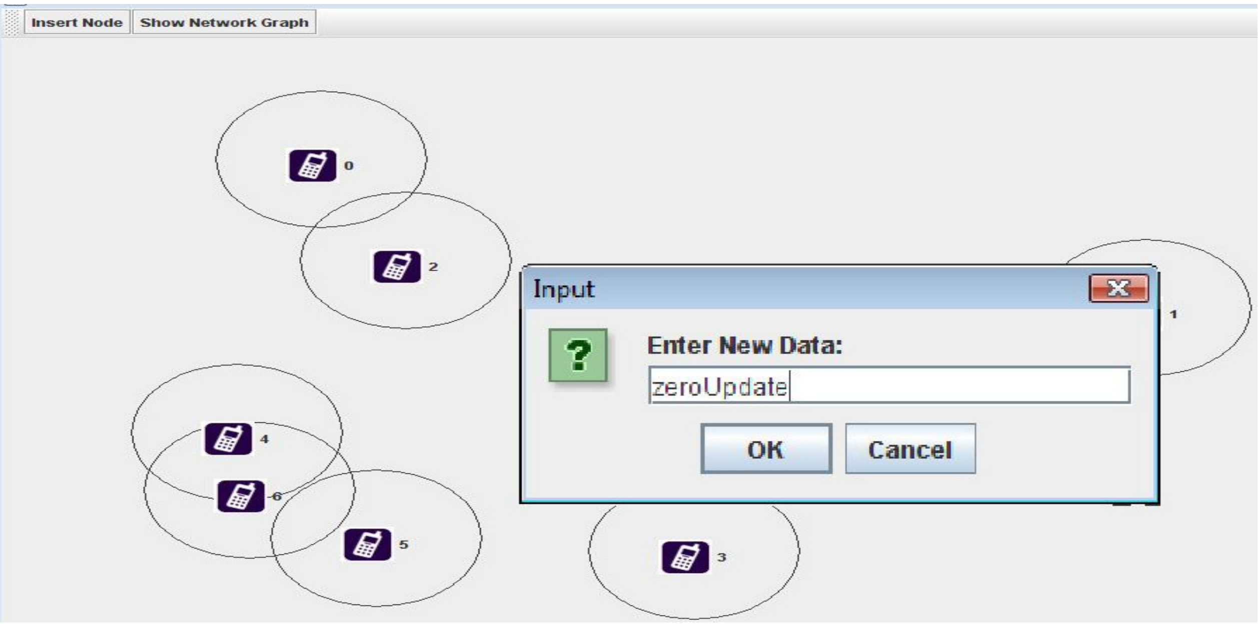


FIGURE 11: Updating data item on host

Figure 11 shows entering of new data in place of existing data into the data items list of particular host and the time stamp also updated.

4. CONCLUSION

The proposed replica allocation mechanism in ad hoc networks is a protocol for automatic replication and migration of objects in response to demand changes. In this paper, we have discussed replica allocation in ad hoc networks to improve data accessibility where mobile hosts frequently disconnected due to unstable radio links. We first provide solution for ad hoc replication problem. Later in this paper we designed an algorithm to update a data item in two ways, one way is to update on primary host and another way is to update on nearest neighbor. We also consider minimum network transfer cost due to data items movement. We can say that the cost incurred in reading and updating the data items depends primarily on the objects size, access frequency of the mobile hosts on the data items and the cost associated with the channel between the communicating hosts. This has to be proved through simulations. The derived mechanism is general, flexible and adaptable to cater for various applications in ad hoc networks. This approach is efficient in both execution time and solution quality.

In addition, the replica allocation algorithms in our proposed methods can be modified to apply them in a much larger scale network because they require message exchanges among all connected mobile hosts and relocate replicas.

5. REFERENCES

[1] C K Toh, Ad Hoc Mobile Wireless Networks, Prentice Hall Publishers, 2002.

[2] N.Saxena, G. Tsudik, J.H.Yi, “Efficient node admission for short lived mobile adhoc networks”, 13th IEEE International Conference on Network Protocols, 2005, Digital Object Identifier: 10.1109/ICNP.2005.14, Page(s): 10 pp. – 278.

[3] Behzad, I. Rubin, “High Transmission Power Increases the Capacity of Ad Hoc Wireless Networks”, IEEE Transactions On Wireless Communications, Vol. 5,No. 1, JANUARY 2006.

[4] Y.Qiu, P.Marbach “Bandwidth Allocation in Ad Hoc Networks: A Price-Based Approach”, IEEE INFOCOM 2003.

[5] D.B.Johnson “Routing in ad hoc networks of mobile hosts”**,** Workshop on Mobile Computing Systems and Applications”, 1994, pp.158-163.

[6] B.W.On, M.S.Park “A Reliable Multicast Protocol in Networks with Mobile Hosts”, International Symposium on Distributed Objects and Applications, 2000, pp.27-35.

[7] S.F.Wu, C.Perkins P.Bhagwat, “Caching location data in mobile networking”, IEEE Workshop on Advances in Parallel and Distributed Systems, 1993,pp:71-76.

[8] D. Barbara, T. Imielinski, “Sleepers and Workaholics: Caching Strategies in Mobile Environments”, in ACM SIGMOD, 1994, pp. 1-12.

[9] Y. Huang, P. Sistla, O. Wolfson, “Data Replication for Mobile Computer”, in ACM SIGMOD, 1994, pp. 13-24.

[10] K. L.Wu, P. S. Yu and M. S. Chen, “Energy-efficient caching for Wireless Mobile Computing”, in IEEE International Conference on Data Engineering, 1996, pp. 336-343.

[11] V. Grassi, “Prefetching Policies for Energy Saving and Latency Reduction in a Wireless Broadcast Data Delivery System”, in ACM International Workshop on Modeling, 2000,pp-77.

[12] S. Saurabh, D. Parkes, “Hard-to-Manipulate VCG-Based Auctions”, Technical Report,

Harvard University, 2004.

[13] K. L.Wu, P. S. Yu and M. S. Chen, “Energy-efficient Caching for Wireless Mobile Computing”, in IEEE International Conference on Data Engineering, 1996, pp. 336-343.

[14] S. Khan and I. Ahmad, “A Powerful Direct Mechanism for Optimal WWW Content Replication”, in 19th IEEE International Parallel and Distributed Processing Symposium, 2005.

[15] J. Jing, A. Elmagarmid, A. Helal and R. Alonso, “Bit-sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments”, Mobile Networks and Applications, 2(2), pp. 115-127, 1997.

[16] E. Pitoura and B. Bhargava, “Maintaining Consistency of Data in Mobile Distributed Environments”, in IEEE International Conference on Distributed Systems, 1995, pp. 404-413.

[17] J. Cay, K. L. Tan and B. C. Ooi, “On Incremental Cache Coherency Schemes in Mobile Computing Environments”, in IEEE International Conference on Data Engineering, 1997, pp. 114-123.

Routing in « Delay Tolerant Networks » (DTN) Improved Routing With Prophet and the Model of “Transfer by Delegation” (Custody Transfer)

El Mastapha Sammou sammouelmastapha@yahoo.fr
Department of Computer Science,
Faculty of Science and Technology,
University Cadi Ayyad,
Marrakech, 40000, Morocco

Abdelmounaim Abdali aabdali5@ gmail.fr
Department of Computer Science,
Faculty of Science and Technology,
University Cadi Ayyad,
Marrakech, 40000, Morocco

Abstract

In this paper, we address the problem of routing in “delay tolerant networks” (DTN). In such networks there is no guarantee of finding a complete communication path connecting the source and the destination at any time, especially when the destination is not in the same region of the source, what makes the traditional routing protocols inefficient in that transmission of the messages between nodes. We propose to combine the routing protocol Prophet and the model of "transfer by delegation" (custody transfer) to improve the routing in DTN network and to exploit the nodes as a common carriers of messages between the network partitioned.

To implement this approach and assess those improvements and changes we developed a DTN simulator.

Simulation examples are illustrated in the article.

Keywords: Routing, Delay Tolerant Networks, DTN, Intermittent network connectivity, Simulator.

1. INTRODUCTION

Delay tolerant networks or networks with intermittent connectivity networks are wireless mobile ad hoc often where a communication path between a source node and destination node does not exist, either directly or through established routes by intermediate nodes. This situation occurs if the network is sparse and partitioned into several areas due to high mobility, low density nodes or when the network extends over long distances; In these cases, the traditional routing protocols have been developed for mobile ad hoc networks proved to be insufficient because they require the existence of a dense and connected in order to route the packets, To resolve this problem of routing in DTN networks, researchers have proposed the use of routing approaches based on the Principe "Store-Carry-and-forward [8], such as:

The epidemic routing protocol [9]: Messages propagate through the network like an outbreak of disease. This approach ensures that the message reaches its destination as much as possible, but it also wastes a lot of resources by unnecessary transfers of messages.

The Prophet routing protocol [1] is one of the routing algorithms that have been proposed to use these resources properly. Prophet introduced a metric called Delivery Predectability , $P(A, B) \in [0, 1]$. This metric is calculated by each node A of the DTN network and for each known destination B and will be used to decide which messages to be exchanged whenever two nodes meet.
The model of "transfer by delegation"(custody transfer) [2] [3] [8]: In this model by assigning responsibility for a message to a single node at any time. This model has the advantage of being economical in terms of resources, since a message is the responsibility of a node at any time during its delivery. However it now risk losing the message if the wearer goes down or destroyed.

Our approach is to combine the two approaches to routing, Prophet [1] and the model of "transfer by delegation" (custody transfer) [2] [3] [8] to overcome the problems of routing in DTN networks.

2. OUR APPROACH TO ROUTING

Normally, one of the most fundamental requirements is to find a communication path between nodes in a sparse network and partitioned into several zones, in this case, communication between areas of the network depends only on the displacement of certain nodes between areas (as shown in Figure 1) As the delivery of messages depends on the mobility of nodes, it is very difficult to obtain global information and routing becomes an important issue.

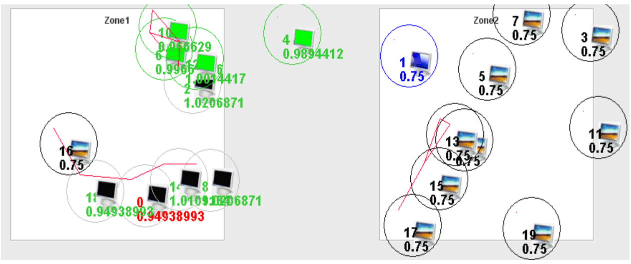


FIGURE 1: Illustrating the transport of messages by a mobile node moving between two areas, each consisting of a few nodes

With the aim to maximize the chances that a message reaches its destination and to minimize the resources consumed in the network such as bandwidth, capacity of storage devices and the energy of the different nodes in an environment characterized by disconnections that often occur because of the low density of nodes, node mobility and energy failure.

Our approach is to combine the routing protocol Prophet [1] and the model of "transfer by delegation"(custody transfer) [2] [3] [8] to exploit the nodes as carriers of messages between the network partitioned . The combination of these two approaches (Prophet and The model of "transfer by delegation") combines two kinds of routing technique based on the degree of the knowledge that the node has about its future contacts with other nodes in the network [4]:

- Technique of controlled routing.
- Technical routing predicted.

The key issues resolved by our approach:

- The choice of nodes that can act as carriers of messages (delegates) between the network partitioned.
- Nodes incorporating elements of knowledge and contextual elements.
- Increases the chances that a message reaches its destination while minimizing the time from End to End.
- Economic from the point of view of the network resources consumed.

In this work we have developed a DTN simulator written in Java which is based on our approach to evaluate the different routing parameters.

3. SCENARIO OF OUR APPROACH

The probability of delivery are calculated locally in each area according to the approach of Prophet [1] and the nodes move according to the two mobility models: Model *Random Waypoint* and model *Restricted Random Waypoint* [10] [11] [12] as shown in Figure 2:

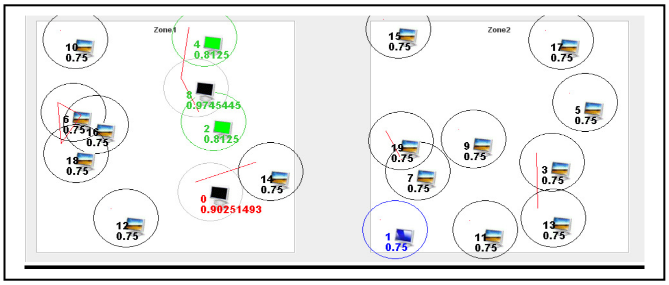


FIGURE 2: Illustrates the calculation of probability according to Prophet in each zone.

N0 node wants to send a message to N1. This can not be done because there is no path between the two areas. The message is sent to N4 that has a better probability of delivery and a planned movement and stores it. As shown in Figure 3:

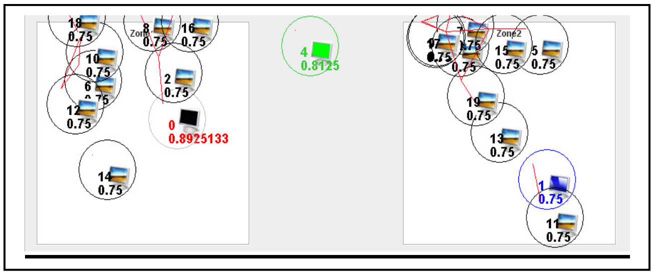


FIGURE 3: Illustrates the transport of messages by a mobile node moving between two areas.

After a certain period of time, N4 moves to another area (as shown in Figure 4). The message reached its intended recipient using the routing protocol Prophet.

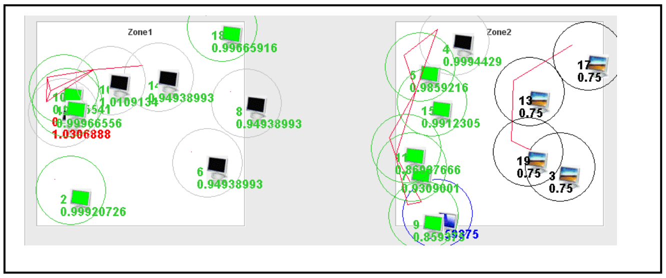


FIGURE 4: Illustrates the node N4 in the second zone so that the message is delivered to its intended recipient

4. STRATEGY FOR TRANSMISSION OF OUR APPROACH

When the source and destination are in two different areas [1] [2] [3] [6] [7] [8]

1. Nodes that can act as carriers are the nodes that have:
   - A high probability of delivery.
   - The planned movements between zones.
   - A sufficient transmission energy.
2. In the case of a network where there are nodes that have random movements, the best carriers are the nodes that have:
   - A high probability of delivery.
   - A sufficient transmission energy.
3. In the case of a network where there are nodes that have random movements and nodes which have planned movements, the best carriers are the nodes that have:
   - A planned and controlled movement between areas.
   - A sufficient transmission energy.
4. When the nodes which have planned and controlled movements between zones are nodes that have better features in terms of energy and storage capacity as the case of buses, planes, trains … We may use them :
   - On the one hand as the best carriers (delegates) of the message.
   - Secondly as fixed relay " mobile "with a periodic occurrence in both areas, these relays can be exploited on the one hand to describe the movements and mobility of nodes [5] in each zone, based on the frequency of visits to these relays [6], secondly to increase the number of contacts between nodes.

The principle of communication in the same area [1] [2] [3] [6] [7] [8]

- When a node encounters another node with the greatest probability of delivery, it sends the message to that node and still keep the message for transmission to other nodes in the future.
- When a node encounters another node that has a planned movement and a low probability of delivery, it sends this message to the node even if the probability of delivery is low, then it deletes the copy of the message, then it frees up the space at its storage unit.
- When a node encounters another node that has a planned movement and a high probability of delivery, it sends the message to this node, then it deletes the copy of the message, then it releases the space at its storage unit.

Mechanisms of acquittals

The acknowledgment mechanism between nodes is done according to the acknowledgment mechanism used by the model of "transfer by delegation"(custody transfer) [2] [3] [8].

Cases where the nodes are not allowed to transmit messages

- When a node encounters another node that has a high probability of delivery and has not enough energy.
- When a node encounters another node that has a planned movement and controlled and not enough energy.
- When a node encounters another node that has a planned and controlled movement and a high probability of delivery and has not enough energy.

5. SIMULATION AND TEST

The simulator is written in Java. JAVA is an object-oriented programming language, allows one hand to develop real applications and the other hand the object-oriented approach considers a program as consisting of a set of objects which adapts our approach. Figure 5 shows the general design of the application and Figure 6 shows the main interface of the application.

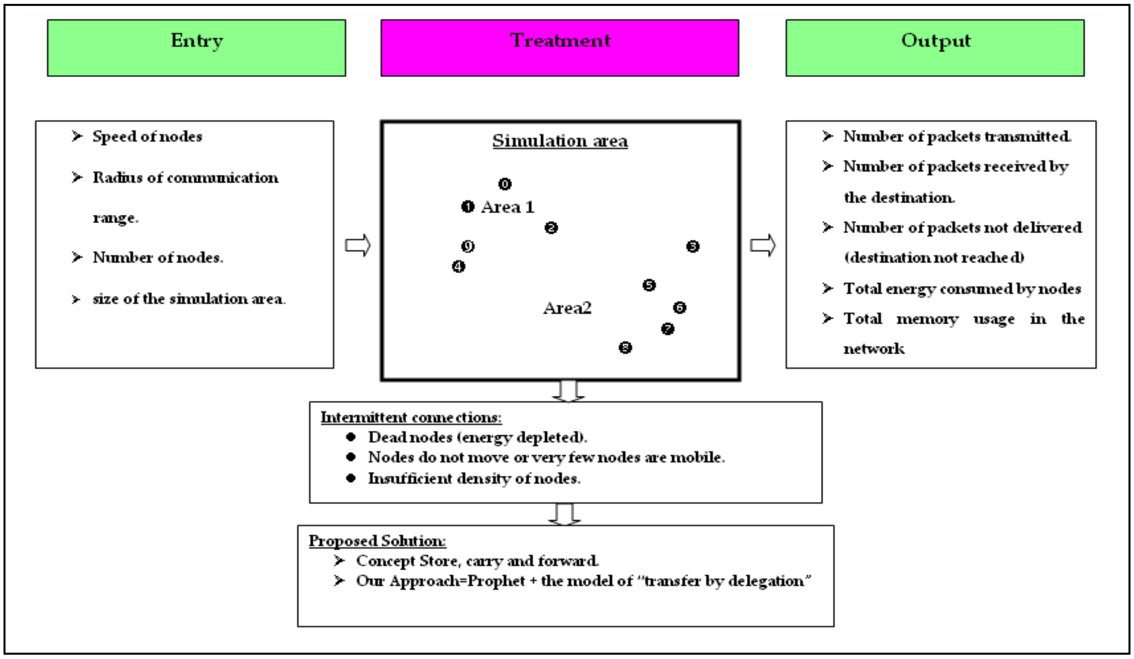
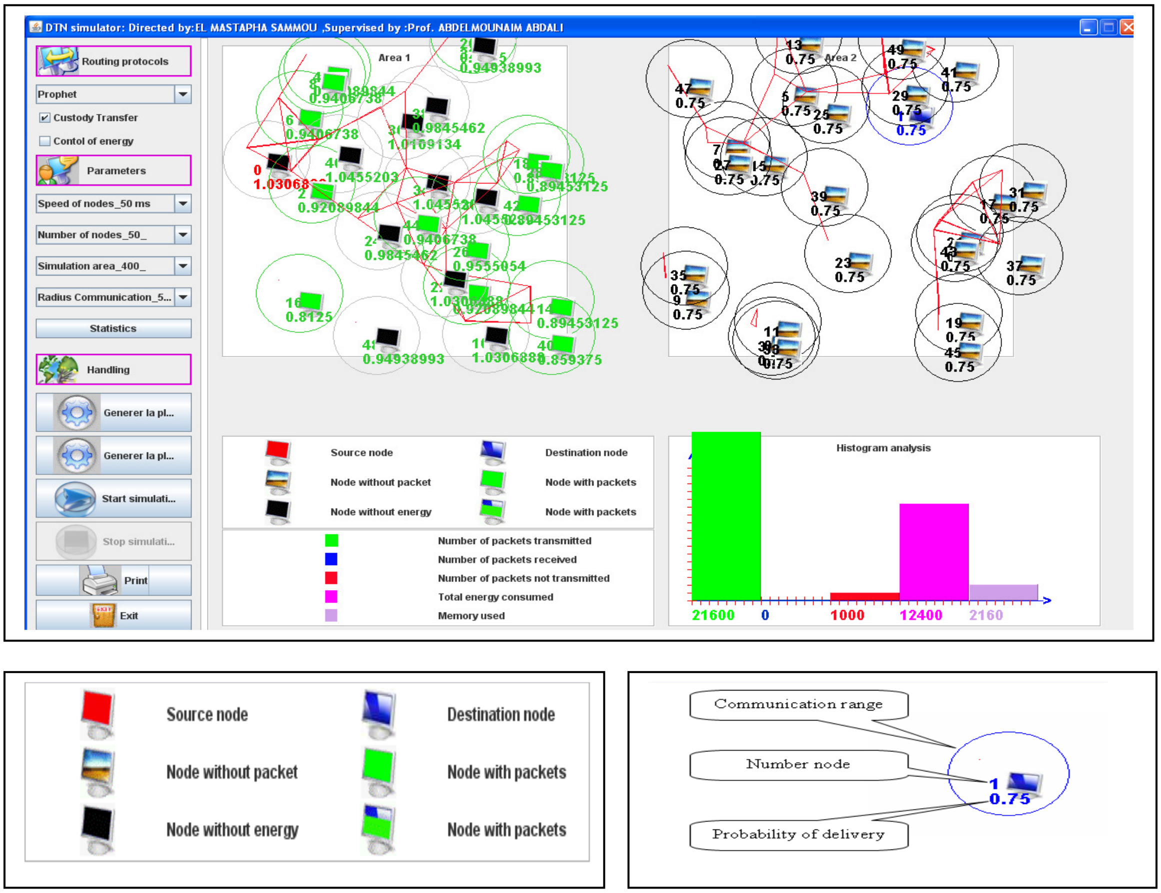

Entry
Treatment
Output
Speed of nodes
Radius of communication range.
Number of nodes.
size of the simulation area.
Simulation area
Area 1
Area2
Number of packets transmitted.
Number of packets received by the destination.
Number of packets not delivered (destination not reached)
Total energy consumed by nodes
Total memory usage in the network
Intermittent connections:
Dead nodes (energy depleted).
Nodes do not move or very few nodes are mobile.
Insufficient density of nodes.
Proposed Solution:
Concept Store, carry and forward.
Our Approach=Prophet + the model of "transfer by delegation"

FIGURE 5: Illustrates the general design of the application


DTN simulator: Directed by:EL MASTAPHA SAMMOU ,Supervised by :Prof. ABDELMOUNAIM ABDALI

Routing protocols

Prophet

☑ Custody Transfer

□ Contol of energy

Parameters

Speed of nodes_50 ms

Number of nodes_50_

Simulation area_400_

Radius Communication_5...

Statistics

Handling

Generer la pl...

Generer la pl...

Start simulati...

Stop simulati...

Print

Exit

🟥	Source node	🟦	Destination node
	Node without packet	🟩	Node with packets
■	Node without energy		Node with packets

🟩	Number of packets transmitted
🟦	Number of packets received
🟥	Number of packets not transmitted
🟪	Total energy consumed
🟪	Memory used

🟥	Source node	🟦	Destination node
	Node without packet	🟩	Node with packets
■	Node without energy		Node with packets

FIGURE 6: Illustrates the main interface of the application

Assumptions and Data Analysis

1. Assumptions: We assume that Packet routing is done from the "store-carry-and-forward" ; The volume of the queue is considered infinite; The nodes move according to the two models of random mobility: Restricted random waypoint and Radom Waypoint [10] [11] [12]; The speed of nodes varies between Vmin= 200 ms and Vmax = 50 ms; The space of traveled nodes varies between 50 and 500 units of surface; The number of nodes varies between 8 and 300 nodes; The number of packets from the source is 1000 packets; The energy level of each node is 1000 units of energy ; The probability of delivery is calculated locally according to formulas presented by the routing protocol prophet [1].

$$P(A,B) = P(A,\mathrm{B})_{old} + (1 - P(A,B)_{old}) * P_{init}$$
$$P(A,C) = P(A,C)_{old} + (1 - P(A,C)_{old}) * P(A,B) * P(B,C) * \beta$$
$$P(A,B) = P(A,B) * \Upsilon^{k}$$

2. Data analysis: In this simulator we have analyzed the following data: The number of packets transmitted in the network; The number of packets not sent; The number of packets received by the destination; The energy consumed in the network; The amount of memory consumed in the network. This analysis is done according to the approach of the Prophet and in our approach to compare the two approaches.

Tests and results

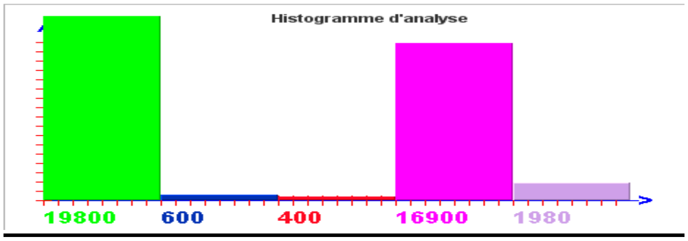
1. **Scénario1**

Both areas are dense and connected. The internal connectivity of each zone is guaranteed. However, there is no permanent connection between the two areas.

Simulation parameters 1	Values
Number of nodes	50
Energy level of each node	1000
Radius of the focused communication	50 m
Maximum Speed	50 ms
Size of each area	300*300
Simulation time	1500 ms
Initial probability	0,75
β	0,25
ϒ	0,98

TABLE1: Parameters of simulation 1

- Simulation results under the approach of Prophet.



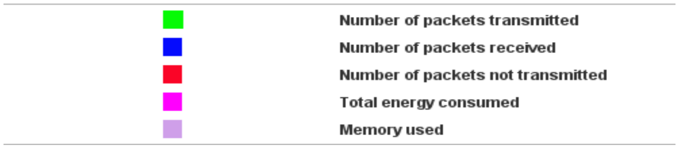


FIGURE 7: Illustrates the result of analysis of the approach of the prophet depending on the parameters of simulation 1

- Simulati

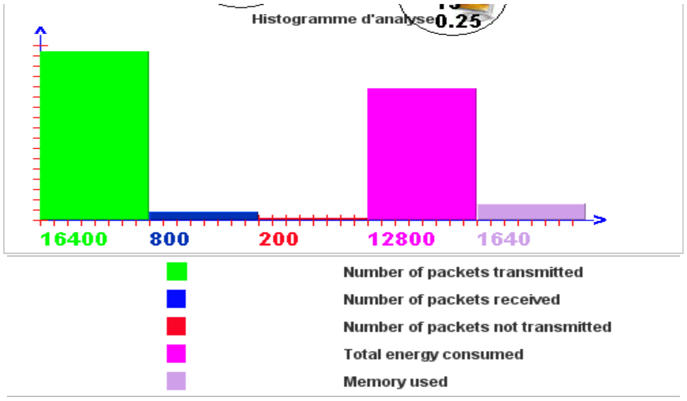


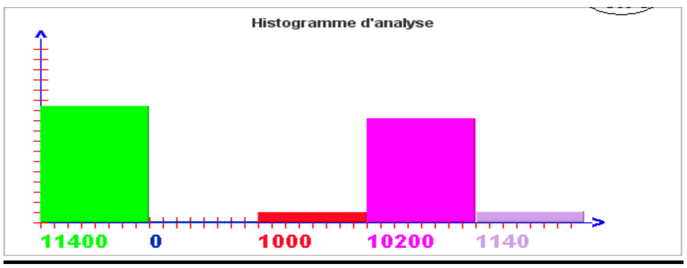
Figure 8: Illustrates the analytical result of our approach Depending on the parameters of simulation 1

2. **Scénario2**: The node density is low in both areas. The internal connectivity of each zone is not guaranteed and there is no permanent connection between the two areas.

Simulation parameters 2	Values
Number of nodes	20
Energy level of each node	1000
Radius of the focused communication	40
Maximum Speed	50 ms
Size of each area	500*500
Simulation time	1500 ms
Initial probability	0,75
β	0,25
ϒ	0,98

TABLE2: Parameters of simulation 2

- Simulation results under the approach of Prophet.



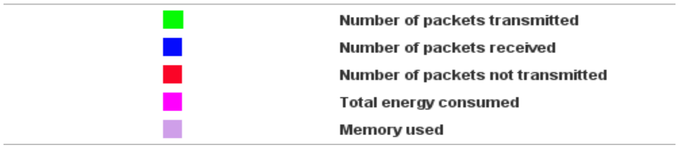


FIGURE 9: Illustrates the result of analysis of the approach of Prophet depending on the parameters simulation 2

- Simulation results based on our approach.

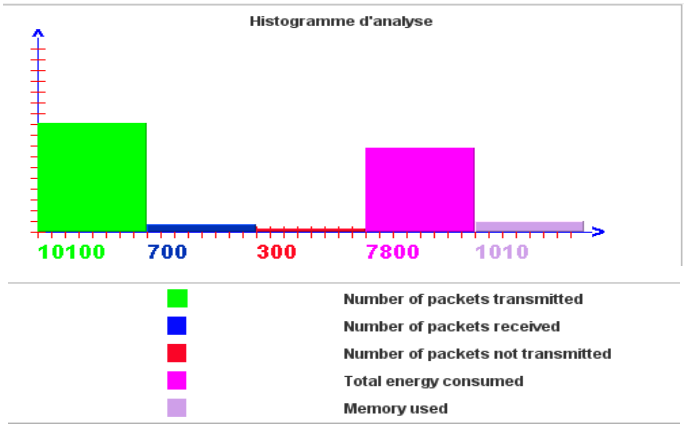


FIGURE 10: Illustrates the result of analysis of our approach Depending on the parameters of simulation 2

6. CONCLUSION

DTN networks suffer from several shortcomings related to routing, especially when the network is partitioned into several zones and where the destination is not in the same region of the source, what makes traditional routing protocols ineffective to the extent of transmit messages between nodes.

We proposed an approach that involves combining the routing protocol Prophet and the model of "transfer by delegation" (custody transfer) to improve routing in DTN networks based on contextual elements and the elements of knowledge a node has about its future contacts with other nodes in the network.

According to the simulations realized, our approach has good performance in comparison to the Prophet algorithm, but its effectiveness can be further improved.

7. REFERENCES

1. A.Lindgren, A. Doria and O. Scheln. *Probabilistic routing in intermittently connected networks. In Proceedings of ACM MobiHoc (poster session), Maryland, USA, June2003.*

2. F. GUIDEC, *"Deployment and implementation support services communicating in pervasive computing environments, ", "Déploiement et support à l'exécution de services communicants dans les environnements d'informatique ambiante,"L'UNIVERSITÉ DE BRETAGNE SUD, June 2008, pp. 35-65*

3. Kevin Fall, Wei Hong, and Samuel Madden. *Custody Transfer for Reliable Delivery in Delay Tolerant Networks. Technical report, Intel Research Berkeley, 2003.*

4. M. IBRAHIM *“Routing and performance evaluation of Disruption tolerant networks,” l’Université de Nice - Sophia Antipolis, Novembre 2008.*

5. *J. Leguay, “Heterogeneity and Routing in Delay Tolerant Networks,” l’Université Paris VI, juillet 2007.*

6. J. Leguay, T. Friedman and V. Conan*, “ Evaluating Mobility Pattern Space Routing for DTNs,” Université Pierre et Marie Curie, Laboratoire LiP6–CNRS.*

7. M. Musolesi, S. Hailes and C. Mascolo*, “Context-aware Adaptive Routingfor Delay Tolerant Mobile Networks” in Proc. WOWMOM, 2005.*

8. F. Warthman*, “Delay Tolerant Networks”, Delay Tolerant Networking Tutorial,2003, http://www.ipnsig.org/reports/DTN_Tutorial11.pdf*

9. A. Vahdat and D. Becker*, “Epidemic routing for partially connected ad hoc networks,” Technical Report CS-200006, Duke University, April 2000”.*

10. Christian Bettstetter, Hannes Hartenstein*, and Xavier Pérez-Costa, “Stochastic properties of the random waypoint mobility model,” ACM/Kluwer Wireless Networks, Special Issue on Modeling and Analysis of Mobile Networks, 10(5) :555–567, Sept.2004.*

11. Christian Bettstetter, Giovanni Resta, and Paolo Santi*. The node distribution of the random waypoint mobility model for wireless ad hoc networks. IEEE Transactions on Mobile Computing, 2(3) :257–269, July 2003.*

12. Christian Bettstetter and Christian Wagner*, “The spatial node distribution of the random waypoint mobility model,”. In Proc. German Workshop on Mobile Ad-Hoc Networks (WMAN), GI Lecture Notes in Informatics, Ulm, Germany, Mar. 2002.*

Wireless Sensor Network Simulators
A Survey and Comparisons

Harsh Sundani
EECS Department,
University of Toledo,
2801 W. Bancroft, MS 301, Toledo, Ohio 43606

Haoyue Li
EECS Department,
University of Toledo,
2801 W. Bancroft, MS 301, Toledo, Ohio 43606

Vijay Devabhaktuni vijay.devabhktuni@utoledo.edu
EECS Department,
University of Toledo,
2801 W. Bancroft, MS 301, Toledo, Ohio 43606

Mansoor Alam
EECS Department,
University of Toledo,
2801 W. Bancroft, MS 301, Toledo, Ohio 43606

Prabir Bhattacharya
Department of Computer Science,
University of Cincinnati,
814 Rhodes Hall, Cincinnati, OH 45221

Abstract

Simulation tools for wireless sensor networks are increasingly being used to study sensor webs and to test new applications and protocols in this evolving research field. There is always an overriding concern when using simulation that the results may not reflect accurate behavior. It is therefore essential to know the strengths and weaknesses of these simulators. This paper provides a comprehensive survey and comparisons of various popular sensor network simulators with a view to help researchers choose the best simulator available for a particular application environment. It also provides a detailed comparison describing the pros and cons of each simulator.

Keywords: Wireless sensor networks, Simulator, Emulator, Comparison, Performance evaluation.

1. INTRODUCTION

Sensor networks are composed of large numbers of tiny sensing and computing devices. Each of these devices, called motes, has very limited communication, computational and energy resources. Often embedded in uncontrolled physical environments, these networks require distributed algorithms for efficient data processing, while individual motes require highly concurrent and reactive behavior for efficient operation. Sensor networks face many problems that do not arise in other types of networks [1]. Power constraints, limited hardware, decreased reliability, and a typically higher density and number of nodes than those found in conventional networks are few of the problems that have to be considered when developing protocols for use in sensor networks.

Fig.1 shows a typical simple wireless sensor network. As can be seen, a complete wireless sensor network usually consists of one or more base stations (or gateway), a number of sensor nodes, and the end user. Sensor nodes are used to measure physical quantities such as temperature, position, humidity, pressure etc. The output of those sensor nodes are wirelessly transmitted to the base station (or gateway) for data collection, analysis, and logging. End users may also be able to receive and manage the data from the sensor via a website from long-distance or applications in console terminal [2]. However due to the associated cost, time and complexity involved in implementation of such networks, developers prefer to have first-hand information on feasibility and reflectivity crucial to the implementation of the system prior to the hardware implementation.

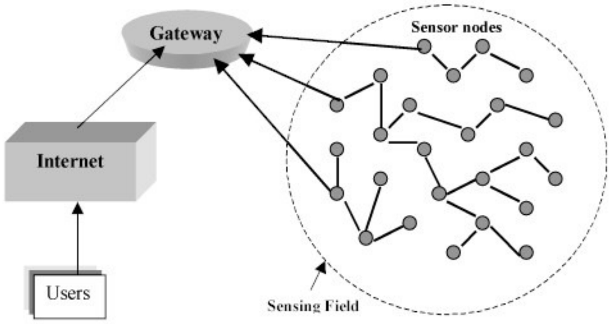


FIGURE 1: A simple wireless sensor network

This is especially true in sensor networks, where hardware may have to be purchased in large quantities and at high cost. Even with readily available sensor nodes, testing the network in the desired environment can be a time consuming and difficult task. Simulation-based testing can help to indicate whether or not the time and monetary investments are worthwhile. Simulation is, therefore, the most common approach to developing and testing new protocol for sensor networks. There are a number of advantages to this approach including lower cost, ease of implementation, and practicality of testing large-scale networks.

In order to effectively develop any protocol based on simulations, it is important to know the different tools available and their benefits and drawbacks. Given the facts that simulation is not perfect and that there are a number of popular sensor network simulators available, thus making different simulators accurate and most effective for different situations/applications. It is crucial for a developer to choose a simulator that best fits the application [3]. However, without a working knowledge of the available simulators, this is can be a challenging task. Additionally, knowing the weaknesses of available simulators could help developers to identify drawbacks of their own models, when compared with these simulators, thus providing an opportunity for improvement. It is thus imperative to have a detailed description of a number of the more prominent simulators available. In this paper, we have compared various sensor network simulators with emphasis on their ease of use, key features, limitations, availability, and environments best supported [4].

2. DESIGN OF SENSOR NETWORK SIMULATOR

The design of a Wireless Sensor Network (WSN) is a very application-specific task, especially because of the peculiarity of the considered deployment environment. Generic reliable predictive models for data correlation or radio propagation are seldom available. A thorough preliminary test phase is thus necessary, either by means of specifically crafted test beds, or via reliable simulations. WSN applications must be tested on a large scale, and under complex and varying conditions in order to capture a sufficiently wide range of interactions, both among nodes, and with the environment. A WSN simulator consists of various modules namely events, medium, environment, node, transceiver, protocols, and applications. Each category is represented by an interface that defines its methods and events generated and consumed.

1. Event
Event is an abstract base class that provides basic functionality for all events. It contains the time at which an event should work, and provides methods to: compare events based on their fire times, determine whether events are equal, print themselves to a string, and an abstract method to fire the event.

2. Medium
Medium models the wireless medium. It allows nodes to broadcast signals, and is responsible for informing nodes of signals that affect it. In order to do this, Medium must be informed of the presence of every node, and any changes in position or radio properties such as transmitter power or receiver sensitivity. Medium has the properties of bandwidth and wavelength of the medium modeled and a reference to a propagation model that is given to it at the time of construction. The propagation model provides the strength at a particular receiver from a signal transmitted by a given transmitter.

3. Environment
The Environment module is similar to Medium module. The difference is that the implementation of Environment has properties that relate to the physical phenomenon modeled. Environment also has a propagation model that models the propagation of the physical phenomena modeled. Physical phenomena of interest in sensor networks include: temperature, light, humidity, magnetic field, sound, optical, chemical presence.

4. Node
It represents a single node in a wireless sensor network. As such, it serves as a container for all of the components, both hardware and software, in a node. These components should be included: processor, transceiver, sensors, actuators, energy source (such as a battery), network protocols, and applications. In addition each node has the properties of location and identification.

5. Transceiver
Transceiver models the hardware transceiver on each sensor node. It models the transceiver states (i.e. sleep, standby, receive, and transmit), and their associated behavior and power consumption. Transceiver consumes events informing it of the beginning and ending of every signal it receives. It sums active signals to maintain the interference. Transceiver generates events for the beginning and ending of every signal it transmits. These events are all exchanged with an instance of the Medium module.

6. Physical Protocol
The Physical protocol is the lowest layer in a network stack. It is often implemented in the transceiver hardware. The Physical layer provides services for: changing the state of the transceiver, carrier sensing, sending and receiving packets, received energy detection on received packets, changing channels on physical layers that support multiple channels.

7. MAC Protocol
The MAC protocol is the next layer in a network stack. It is usually implemented in software running on the node’s processor. The MAC layer provides services for: changing the state of the MAC layer (i.e. low power mode), setting and getting protocol parameters, sending and receiving packets, etc. A WSN simulator usually offers implementations for several sensor network MAC protocols.

8. Routing Protocol
The Routing protocol resides above the MAC protocol and provides services for routing messages over multiple hops between nodes that cannot communicate directly.

9. Application Layer

The Application layer resides at the top of the network stack. It interfaces with the lower layers in the network stack as well as the sensors and actuators to implement a wireless sensor network application.

Most of the WSN simulators are based on the design described above. In addition to including the different modules, a WSN simulator should also have the following capabilities:

i. Reusability and availability
Simulation is used to test novel techniques in realistic and controlled scenarios. Researchers are usually interested in comparing the performance of a new technique against existing proposals [5].

ii. Performance and scalability
Performance and scalability is a major concern when facing WSN simulation. The former is usually bounded to the programming language effectiveness. The latter is constrained to the memory, processor and logs storage size requirements [6].

iii. Support for rich-semantics scripting languages to define experiments and process results
The vast amount of variables involved in the definition of a WSN experiment requires the use of specific input scripting languages, with high-level semantics. Additionally, it is likely that large quantities of output data will also be generated through many replicas of the experiments [7]. Therefore, a suitable output scripting language, which helps to obtain the results from the experiments quickly and precisely is desirable.

iv. Graphical, debug and trace support.
Graphical support for simulations is interesting in three aspects:
(a) As a debugging aid. The primary and more practical way to quickly detect a bad behavior is to “watch” and follow the execution of a simulation. The key features that a graphical interface should support are: Capability of inspection of modules, variables and event queues at real time, together with “step-by-step” and “run-until” execution possibilities. These features make graphical interfaces a very powerful debugging tool. Note that the key is the ability to interact with the simulation.
(b) As a visual modeling and composition tool. This feature usually facilitates and speeds the design of small experiments or the composition of basic modules. However, for large scale simulations, it is not very practical.
(c) Finally, it allows quick visualization of results without a post-processing application [8].

However, there are various challenges associated with the available WSN simulators. For instance, some simulator lack of available protocol models, which causes the increase of developing time, some simulators limit the scalability, etc. Additionally, modeling problems arise when considering the new environment and the energy components. They also compromise scalability and accuracy [9]. A deep study of these issues is mandatory for a better understanding and characterization of sensor networks and their corresponding simulators.

3. SIMULATOR COMPARISON

There are many different possible platforms for simulation and testing of routing protocols for WSNs. Below is a list of some of the most popularly used simulators for WSNs. Different aspects like energy efficiency, resources, decentralized collaboration, fault tolerance, simulation scenarios, global behavior etc. have been compared.

3.1 NS-2

1. Summary: NS-2 is a discrete event simulator targeted at networking research. NS-2 began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. NS-2 has a modular approach and hence is effectively extensible [10].
2. Environment: It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Support for wireless networks was added later to simulate wireless LAN protocols, mobile ad-hoc networks and wireless sensor networks. The simulator focuses on following the ISO/OSI model.

3. Simulation language: Simulations are based on a combination of C++ and OTcl. [11].

4. Key features: NS-2's extensibility is perhaps what has made it so popular for sensor networks. It has an object-oriented design which allows for straightforward creation and use of new protocols. The key features for WSNs include sensor channels, battery models, lightweight protocol stacks, hybrid simulation support, and scenario generation tools. It provides a visualization tool called NAM (Network AniMator). Due to its popularity and ease of protocol development, there are a high number of different protocols that are publicly available. All the simulations are run at the packet level, allowing for detailed results.

5. Limitations: NS-2 has a long learning curve and requires advanced skills to perform meaningful and repeatable simulations. Another drawback to NS-2 is the lack of customization available. Packet formats, energy models, MAC protocols, and the sensing hardware models all differ from those found in most sensors. NS-2 also lacks an application model. In many network environments this is not a problem, but sensor networks often contain interactions between the application level and the network protocol level.

3.2 TOSSIM

6. Summary: TOSSIM is a discrete event simulator for TinyOS wireless sensor networks, which were developed at UC Berkeley. TOSSIM captures the behavior and interactions of networks not on the packet level but at network bit granularity. TOSSIM is designed specifically for TinyOS applications to be run on MICA Motes. TOSSIM simulates entire TinyOS applications [12]. It works by replacing components with simulation implementations. TOSSIM can replace a packet-level communication component for packet-level simulation, or replace a low-level radio chip component for a more precise simulation of the code execution [13] [14].

7. Environment: TinyOS is a sensor network operating system that runs on custom 'mote' hardware. TOSSIM provides mechanisms for TinyOS developers to choose the accuracy and complexity of the radio model necessary for their simulations [15].

8. Simulation language: It has a component-based programming model, provided by the nesC language, a dialect of C [16].

9. Key features: TOSSIM simulates the TinyOS network stack at the bit level, allowing experimentation with low-level protocols in addition to top-level application systems [17]. The simulation provides several mechanisms for interacting with the network, packet traffic can be monitored and packets can be statically or dynamically injected into the network. The transmission is simulated at the bit level.

10. Limitations: TOSSIM's run-instantly execution model does not capture CPU time. Since interrupts are discrete events, TOSSIM does not model preemption and the resulting possible TinyOS data races. Compilation steps lose the fine-grained timing and interrupt properties of the code, which can be important when the application runs on the hardware and interacts with other nodes [18]. TOSSIM is based on the assumption that each node in the network must run the exact same code, thus making it less flexible. TOSSIM does not model energy consumption, though there is an add-on PowerTOSSIMz that corrects this problem.

3.3 GLoMoSim

11. Summary: Global Mobile Information System Simulator (GloMoSim) is a scalable simulation environment for large wireless and wired communication networks. The node aggregation technique is introduced into GloMoSim to give significant benefits to the simulation performance. In GloMoSim, each node represents a geographical area of the simulation. Hence the network nodes which a particular entity represents are determined by the physical position of the nodes.

12. Environment: GloMoSim has several choices for radio propagation, CSMA MAC protocols (including 802.11), mobile wireless routing protocols, and implementations of UDP and TCP. GloMoSim is good at simulating mobile IP networks [19].

13. Simulation language: GloMoSim is a library for the C-based parallel discrete-event simulation language PARSEC (Parallel Simulation Environment for Complex Systems)

14. Key features: The ability to use GloMoSim in a parallel environment distinguishes it from most other sensor network simulators. Like Ns-2, GloMoSim is also designed to be extensible, with all protocols implemented as modules in the GloMoSim library [20]. GloMoSim can be executed using a variety of synchronization protocols and was successfully implemented on both shared memory and distributed memory computers.

15. Limitations: GloMoSim currently supports protocols for a purely wireless network. In the future, the developers anticipate adding functionality to simulate a wired as well as a hybrid network with both wired and wireless capabilities. GloMoSim is effectively limited to IP networks because of low level design assumptions. Therefore, it suffers the same problems as Ns-2, the packet formats, energy models, and MAC protocols are not representative of those used in wireless sensor networks.

3.4 UWSim

16. Summary: UWSim is a simulator used for Underwater Sensor Networks (UWSN). Most of the currently available simulators focus greatly on ground-based sensor and ad hoc networks they do not consider the factors that affect the underwater communication [21]. UWSim, on the other hand, has its focus on handling scenarios specific to UWSN environments, for example low bandwidth, low frequency, high transmission power, and limited memory. It is based on a network component-based approach, rather than a layer/protocol-based approach.

17. Environment: Tailor-designed for simulations of under-water sensor networks.

18. Simulation language: The software was developed on Windows XP using Microsoft. Net Framework 2.0 and is made to support all versions of Windows including 64-bit machines. The actual software development was carried out in a purely object-oriented fashion using C# capabilities [22].

19. Key features: Most simulators assume the characteristics of ground-based wireless ad hoc and sensor networks. These simulators use radio frequency transmission whereas UWSN needs a simulator which simulates the acoustic network. UWSim simulates the acoustic network [23]. It is based on a novel routing protocol proposed by the developers, unlike traditional simulators which are based on either proactive or reactive routing protocols such as AODV and DSR. The various characteristics of underwater networks such as low bandwidth, need for high frequency and the effect of salinity and temperature with depth are considered while simulating sensor networks with UWSim.

20. Limitations: Currently, UWSim has support for limited number of functionalities and it calls for further extensions to support a wide range of UWSN simulation exercises. Also another obvious disadvantage is that it cannot be used for simulating any other sensor network except UWSN.

3.5 Avrora

21. Summary: Avrora, a research project of the UCLA Compilers Group, is an open-source cycle-accurate simulator for embedded sensing programs. Unlike other simulators, that are able to simulate only specific platforms, Avrora has language and operating system independence. It provides a framework for program analysis, allowing static checking of embedded software and an infrastructure for future program analysis research. Avrora simulates a network of motes, runs the actual microcontroller programs (rather than models of the software), and runs accurate simulations of the devices and the radio communication [24].

22. Environment: It can emulate two typical platforms, Mica2 and MicaZ, and run AVR elf-binary or assembly codes for both platforms.

23. Simulation/Programming language: It is implemented in Java, which helps flexibility and portability.

24. Key features: One of the key features of Avrora is that it is an accurate and scalable simulator for the actual hardware platform on which sensor programs run. Avrora has a nearly complete implementation of the mica2 hardware platform, including a nearly complete ATMega128L implementation, and an implementation of the CC1000 AM radio [25]. Avrora is also capable of running a complete sensor network simulation with full timing accuracy, allowing programs to communicate via the radio using the software stack provided in TinyOS. It also has an extension point that allows users to create a new simulation type and choose the type of simulation to perform, depending on the number and orientation of the nodes. Developers claim that Avorora scales to networks of up to 10,000 nodes and performs as much as 20 times faster than most other simulators with equivalent accuracy [26].

25. Limitations: One major limitation of Avrora is that it does not model clock drift, a phenomenon where nodes may run at slightly different clock frequencies over time due to manufacturing tolerances, temperature, and battery performance. In recent literature, the developers have modeled distance-attenuation for multi-hop scenarios, but have not yet been able to model mobility. It is 50% slower than TOSSIM.

3.6 SENS: A Sensor Environment and Network Simulator

26. Summary: SENS is a wireless sensor network simulator with modular, layered architecture with customizable components which model an application, network communication, and the physical environment. It enables realistic simulations, by using values from real sensors to represent the behavior of component implementations. Such behavior includes sound and radio signal strength characteristics and power usage.

27. Environment: SENS is a customizable sensor network simulator for WSN applications. Multiple component implementations in SENS offer varying degrees of realism. Users can assemble application-specific environments, such environments are modeled in SENS by their different signal propagation characteristics. The same source code that is executed on simulated sensor nodes in SENS may also be deployed on actual sensor nodes, this enables application portability

28. Simulation/Programming language: It is written in C++. There exists a thin compatibility layer to enable direct portability between SENS and real sensor nodes.

29. Key features: SENS is platform-independent: as new WSN platforms are introduced, their parameter profiles can be added to the simulator. The ability to develop portable applications is an important feature, considering that WSN platforms constantly evolve as new sensor node implementations emerge. Another salient feature of SENS is its novel mechanism for modeling physical environments. WSN applications feature tight integration of computation, communication and interaction with the physical environment. To provide users with the flexibility of modeling the environment and its interaction with applications at different levels of detail, SENS defines an environment as a grid of interchangeable tiles.

30. Limitations: SENS is less customizable than many other simulators, providing no opportunity to change the MAC protocol, along with other low level network protocols. SENS does appear to use one of the most sophisticated environmental models and implements the use of sensors well. However, the only phenomenon detectable is sound. It can be argued as to whether or not it is better for a simulator to support several phenomena well, or to support one phenomenon extremely well.

3.7 COOJA (COntiki Os JAva)

31. Summary: COOJA is a simulator for the Contiki sensor node operating system. MSPSim can be integrated into COOJA, forming COOJA/MSPSim. It allows simultaneous cross-level simulation at application, operating system and machine code instruction set level [27]. COOJA combines low-level simulation of sensor node hardware and simulation of high-level behavior in a single simulation.

32. Environment: COOJA is flexible and extensible in that all levels of the system can be changed or replaced: sensor node platforms, operating system software, radio transceivers, and radio propagation models. As network communication is central to a WSN, the COOJA Simulator supports adding and using different radio mediums [28].

33. Simulation/Programming language: COOJA is a Java application, all interaction with compiled Contiki code is done through Java Native Interface (JNI).

34. Key features: COOJA is primarily a code level simulator for networks consisting of nodes running Contiki OS. Nodes with different simulated hardware and different on-board software may co-exist in the same simulation. Code level simulation is achieved by compiling Contiki core, user processes and special simulation glue drivers into object code native to the simulator platform, and then executing this object code from COOJA [29]. It is able to execute the Contiki programs in two different ways: either by compiling the program code directly on the host CPU, or compiling it for the MSP430 hardware. It can simulate sensor networks simultaneously at different levels, including the operating system level and the network (application) level.

35. Limitations: However, due to its extendibility, the simulator has relatively low efficiency. Simulating many nodes with several interfaces each requires a lot of calculations, especially when plugins are started and registered as observers to those interfaces. Supports a limited number of simultaneous node types, the simulator has to be restarted once and a while if the number of nodes exceed allowable limit. A test interface GUI is absent, thus making extensive and time-dependent simulations difficult.

3.8 Castalia

36. Summary: Castalia is an application-level simulator for Wireless Sensor Network based on OMNeT++. It can be used to evaluate different platform characteristics for specific applications, since it is highly parametric, and can simulate a wide range of platforms. In Castalia, sensor nodes are implemented as compound modules, consisting of sub-modules that represent, for instance, network stack layers, application, and sensor. Node modules are connected to wireless channel and physical process modules [30].

37. Environment: It is a generic simulator with realistic wireless channel and radio model based on measured data. Since it is based on the OMNeT++ platform, it can be used by researchers and developers who want to test their distributed algorithms and/or protocols in realistic wireless channel and radio models, with a realistic node behavior especially relating to access of the radio.

38. Simulation/Programming language: It is developed in C++ at the National ICT Australia [31].

39. Key features: Features of Castalia include: physical process modeling, sensing device bias and noise, node clock drift, and several MAC and routing protocols implemented. Castalia has a highly tunable Medium Access Control (MAC) protocol and a flexible parametric physical process model. Distinct physical process modules in Castalia represent different sensing devices (e.g. temperature, pressure, light and acceleration). Castalia can consider sensing device noise, bias and node clock drift [32].

40. Limitations: It should be noted that Castalia is not sensor-platform specific. Castalia is meant to provide a generic reliable and realistic framework for the first order validation of an algorithm before moving to implementation on a specific sensor platform. It is not useful if one would like to test code compiled for a specific sensor node platform.

3.9 Shawn

41. Summary: Shawn is a customizable sensor network simulator, it is open source and designed to support large-scale network simulation. Instead of simulating a phenomenon, Shawn is designed to simulate the effect of the phenomenon. It is claimed to provide the highest abstract level and support larger network comparing to other simulators such as ns-2, SENSE, OmNeT++, GloMoSim, and TOSSIM. However, details on simulating wireless sensor networks cannot be found [33].

42. Environment: The simulation environment is the home of the virtual world in which the simulation objects reside. The simulated nodes reside in a single World instance. The nodes themselves serve as a container for so-called processors.

43. Simulation/Programming language: It is written in Java.

44. Key features: Shawn features persistence and decoupling of the simulation environment by introducing the concept of Tags. They attach both persistent and volatile data to individual nodes and the world. They decouple state variables from member variables, thus allowing for an easy implementation of persistence [34]. Another benefit is that parts of a potentially complicated protocol can be replaced without modifying.

45. Limitations: Detailed simulations of issues such as radio propagation properties or low-layer issues are not well considered.

3.10 EmStar

46. Summary: EmStar provides a flexible environment for transitioning between simulation and deployment for iPAQ-class sensor nodes running Linux [35]. Users have three options: running many virtual nodes on a single host with a simulated network, running many virtual nodes on a single host with each virtual node bridged to a real-world one for networking, and running a single real node on a host with a network interface.

47. Environment: EmStar offers both a range of runtime environments, from pure simulation to actual deployment. EmStar was designed to be compatible with two different types of nodes. Like the other emulators, it can be used to develop software for Mica2 motes [36]. It also offers support for developing software for iPAQ based microservers.

48. Simulation/Programming language: Emstar is a Linux-based framework [37].

49. Key features: EmStar provides an option to interface with actual hardware while running simulation. The simulation model is component-based and uses a discrete event model. .It has a half simulation/ half-emulation approach similar to SensorSim's, where software is running on a host machine and interfacing with the actual sensor. This allows for use of the actual communication channel and sensors. EmStar code may be run on a diverse set of execution platforms, each run the same code and use the same configuration files, making it easy for developers to seamlessly iterate among all the modes. It brings together a number of the stronger features of other simulators and emulators. EmStar's use of the component-based model allows for fair scalability.

50. Limitations: EmStar uses a very simple environmental model and network medium. As the purpose is to migrate the code to a real sensor environment, simple environment and network medium abstractions sufficed for the developers. Secondly, the simulator will only run code for the types of nodes that it is designed to work with. Also, if a user is attempting to model a system by progressing through the development cycle, he must either ensure that the hardware configuration being used matches the configuration file, or he must bear in mind that there will be differences and compensate appropriately, if necessary [38]. It does not support parallel simulations and lacks algorithms that are reactive to real dynamics, seen in physically situated sensors.

3.11 JSim

- Summary: J-Sim is a general purpose simulator modeled after Ns-2, developed at the University of Washington by the National Simulation Resource. Unlike Ns-2, however, J-Sim uses the concept of components, replacing the notion that each node should be represented as an object. J-Sim uses three top level components: the target node (which produces stimuli), the sensor node (that reacts to the stimuli), and the sink node (the ultimate destination for stimuli reporting). Each component is broken into different parts and modeled differently within the simulator. The breakdown of each component makes it easy to use different protocols in different simulation runs.
- Environment: J-Sim provides support for sensors and physical phenomena. Energy modeling, with the exception of radio energy consumption, is also appropriately provided for sensor networks [39].
- Simulation/Programming language: J-Sim is similar to ns in that is written in two languages, however, in J-Sim's case they are Java and Jacl, a Java version of Tcl.
- Key features: J-Sim features several improvements on Ns-2 and other simulators. Most importantly, its component based architecture scales better than the object oriented model used by Ns-2 and other simulators. Furthermore, J-Sim features an improved energy model and the ability to simulate the use of sensors for phenomena detection. Like SensorSim, applications may be simulated, and there is support for the connection of real hardware sensors to the simulator [40].
- Limitations: J-Sim is however, relatively complicated to use. While no more complicated than Ns-2, the latter simulator is more popular and, thus, more people are willing to spend the time to learn how to use it. J-Sim, while more scalable than many other simulators, also faces its share of inefficiencies. Java, in general, is arguably less efficient than many other languages. There is also unnecessary overhead in the intercommunication model. The only MAC protocol that can be used is 802.11, a problem that seems to occur in most sensor simulators that are built on top of all-purpose simulators.

3.12 SENSE

- Summary: SENSE was a sensor network simulator developed in 2004.
- Environment: SENSE supports an energy model that is sufficient for wireless sensor networks [41].
- Simulation/Programming language: It is similar to J-Sim in that it is component based, but is written in C++ in order to avoid the perceived inefficiency of Java. SENSE runs on top of COST, a component based discrete event simulator that is written in CompC++, a component extension to C++.
- Key features: The most significant feature of SENSE is its balanced consideration of modeling methodology and simulation efficiency. SENSE is a user- friendly simulator that is also very fast. Unlike object-oriented network simulators, SENSE is based on a novel component-oriented simulation methodology that promotes extensibility and reusability to the maximum degree. At the same time, the simulation efficiency and the issue of scalability are not overlooked.
- Limitations: SENSE is still in its active development phase. Although the core of the simulator has been gradually stabilized, it still lacks a comprehensive set of models and a wide variety of configuration templates for wireless sensor networks. Besides, a visualization tool is desirable which can quickly track down what goes wrong during the simulation. Without such a tool, the output of the simulation is hard to interpret. Visualization can also facilitate the configuration phase by allowing networks to be constructed graphically [42].

3.13 VisualSense

- Summary: Modeling of wireless networks requires sophisticated representation and analysis of communication channels, sensors, ad-hoc networking protocols, localization strategies, media access control protocols, energy consumption in sensor nodes, etc. VisualSense is designed to support a component-based construction of such models.
- Environment: VisualSense provides an accurate and extensible radio model. The radio model is based on a general energy propagation model that can be reused for physical phenomena. VisualSense provides a sound model based on this propagation model that is accurate enough to use for localization.

- Simulation/Programming language: VisualSense is a modeling and simulation framework for wireless sensor networks that build on and leverages Ptolemy II. The extension to Ptolemy consists of a few new Java classes and some XML files [43]. The classes are designed to be sub-classed by model builders for customization, although non-trivial models can also be constructed without writing any Java code.
- Key features: It supports actor-oriented definition of network nodes, wireless communication channels, physical media such as acoustic channels, and wired subsystems. The software architecture consists of a set of base classes for defining channels and sensor nodes, a library of subclasses that provide certain specific channel models and node models, and an extensible visualization framework. Customized channels can be defined by subclassing the WirelessChannel base class and by attaching functionality defined in Ptolemy II models [44]. It is intended to enable the research community to share models of disjoint aspects of the sensor nets problem and to build models that include sophisticated elements from several aspects.
- Limitations: VisualSense, however, it does not provide any protocols above the wireless medium, or any sensor or physical phenomena other than sound.

3.14 (J)Prowler

- Summary: Prowler and (J)Prowler are probabilistic wireless sensor network simulators. Prowler is an event-driven simulator that can be set to operate in either deterministic mode (to produce replicable results while testing the application) or in probabilistic mode (to simulate the nondeterministic nature of the communication channel and the low-level communication protocol of the motes). It can incorporate arbitrary number of motes, on arbitrary (possibly dynamic) topology, and it was designed so that it can easily be embedded into optimization algorithms [45].
- Environment: (J)Prowler is targeted to the Berkeley MICA Mote hardware platform running application built on TinyOS, though it could be modified to simulate more general systems.
- Simulation/Programming language: Prowler is written in Matlab, while JProwler is written in Java.
- Key features: The simulator runs under MATLAB, thus it provides a fast and easy way to prototype applications, and has nice visualization capabilities. The network simulator models the important aspects of all levels of the communication channel and the application. The nondeterministic nature of the radio propagation is characterized by a probabilistic radio channel model. A simplified, but accurate model is used to describe the operation of the Medium Access Control (MAC) layer [46].
- Limitations: (J)Prowler provides an accurate radio model. However, it provides only one MAC protocol, the default MAC protocol of TinyOS

4. ANALYSIS

Programming sensor network systems is and will likely remain a challenging task. A primary goal of simulation platforms is to help alleviate these burdens. For wireless sensor network systems, two features of simulators are extremely valuable: reproducible experimentation and dynamic environment modeling. In addition to providing a fairly comprehensive list of simulators available, this paper also addresses most of the issues facing the developers of sensor networks. At the topmost level, developers must decide whether they want a simulator or an emulator. Each has advantages and disadvantages, and each is appropriate in different situations. The key features and limitations of each of these simulators/emulators are highlighted in Table I below.

Generally, a simulator is more useful when looking at things from a high level. The effect of routing protocols, topology, and data aggregation can be seen best at a top level and would be more appropriate for simulation. Emulation is more useful for fine-tuning and looking at low-level results. Emulators are effective for timing interactions between nodes and for fine tuning network level and sensor algorithms. If the developers decide to build a simulator, another design level decision that must be made is whether to build their simulator on top of an existing general simulator or to create their own model. If development

No.	Simulator	Programming language/ Platform	Key features	Limitations
1	Ns-2	C++	Easy to add new protocols. A large number of protocols available publicly. Availability of a visualization tool.	Supports only two wireless MAC protocols, 802.11, and a single-hop TDMA protocol.
2	TOSSIM	nesC	High degree of accuracy or running the application source code unchanged. Availability of a visualization tool.	Compilation steps lose the fine-grained timing and interrupt properties of the code,
3	GloMoSim	Parsec	Parallel simulation capability. It is tailored specifically for wireless networks. Availability of a visualization tool.	Effectively limited to IP networks because of low level design assumptions. Unavailability of new protocols.
4	UWSim	C++	Publicly available and designed solely for UWSN.	Supports only a limited number of functionalities and calls for extension.
5	Avrora	Java	Can handle networks having up to 10,000 nodes. Enables validation of time-dependent properties of large-scale networks	Fails to model clock drift. 50% slower than TOSSIM. Cannot model mobility.
6	SENS	C++	Platform-independent Users can assemble application-specific environments Defines an environment as a grid of interchangeable tiles.	Not accurately simulate a MAC protocol. Provides support for sensors, actuators, and physical phenomena only for sound.
7	COOJA	Java (Simulations in C)	Concerning both simulated hardware and software. Larger-scale behavior protocols and algorithms can be observed.	Not extremely efficient. Supports a limited number of simultaneous node types. Making extensive and time-dependent simulations difficult..
8	Castalia	C++	Physical process modeling, sensing device bias and noise, node clock drift, and several MAC and routing protocols implemented. Highly tunable MAC protocol and a flexible parametric physical process model.	Not a sensor specific platform. Not useful if one would like to test code compiled for a specific sensor node platform.
9	Shawn	Java	Not limited to the implementation of distributed protocols Can simulate vast networks	Detailed simulations of issues such as radio propagation properties or low-layer issues are not well considered.
10	EmStar	Linux	May be run on a diverse set of execution platforms. Combination of simulator and emulator. EmStar's use of the component-based model allows for fair scalability	Only run code for the types of nodes Does not support parallel simulations. Not as efficient and fast as other frameworks
11	J-Sim	Java	Provides support for energy modeling, with the exception of radio energy consumption Support mobile wireless networks and sensor networks. Component-oriented architecture.	Low efficiency of simulation. The only MAC protocol provided for wireless networks is 802.11. Unnecessary run-time overhead
12	SENSE	C++	Balanced consideration of modeling methodology and simulation efficiency. Memory-efficient, fast, extensible, and reusable.	Not accurate evaluation of WSN research. Lacks a comprehensive set of models Absence of a visualization tool
13	VisualSense	Ptolemy II	Provides an accurate and extensible radio model as well as a sound model that is accurate enough to use for localization.	Does not provide any protocols above the wireless medium, or any sensor or physical phenomena other than sound.
14	(J)Prowler	Matlab/Java	Probabilistic wireless sensor network simulators. (J)Prowler provides an accurate radio model.	Provides only one MAC protocol, the default MAC protocol of TinyOS.

TABLE 1: Key features and limitations of some popular WSN simulators

time is limited or if there is one very specific feature that the developers would like to use that is not available, then it may be best to build on top of an existing simulator. However, if there is available development time and the developers feel that they have a design that would be more effective in terms of scalability, execution speed, features, or another idea, then building a simulator from the base to the top would be most effective. In building a simulator from the bottom up, many choices need to be made. Developers must consider the pros and cons of different programming languages, the means in which simulation is driven (event vs. time based), component-based or object-oriented architecture, the level of complexity of the simulator, features to include and not include, use of parallel execution, ability to interact with real nodes, and other design choices. While design language choices are outside of the scope of this paper, there are some guidelines that stand out upon looking at a number of already existing simulators. Most simulators use a discrete event engine for efficiency. Component-based architectures scale significantly better than object-oriented architectures, but may be more difficult to implement in a modularized way. Defining each sensor as its own object ensures independence amongst the nodes. The ease of swapping in new algorithms for different protocols also appears to be easier in object-oriented designs. However, with careful programming, component based architectures perform better and are more effective. Generally, the level of complexity built into the simulator has a lot to do with the goals of the developers and the time constraints imposed. Using a simple MAC protocol may suffice in most instances, and only providing one saves significant amounts of time. Other design choices are dependent on intended situation, programmer ability, and available design time. The use of a standard configuration file may be a limiting factor in many situations.

Figure 2 presents a comparative graph of the most prominent simulation frameworks according to the criteria of scalability and abstraction level. In this figure, it does not express the maximal feasible network sizes, but reflects the typical application domain.

Regarding availability of models, OMNET++, GloMoSim lack of available protocol models compared to other simulators (specially, NS-2), which increases development time. Attending to the ability to compose models from basic pieces, the component or actor based packages J-Sim offer the maximum flexibility. Tools like Shawn or Avrora allow any, Linux or Java respectively, application to be used in a simulation. This feature greatly increases their possibilities. Specific tools such as TOSSIM are able to simulate real sensor code.

Focusing on the performance, one can expect better performance from C/C++ engines than from their Java counterparts. Obviously, parallel simulations should perform and scale better than sequential ones. For instance, parallel simulators as GloMoSim (whose goal is performance rather than scalability) can simulate up to around 10,000 wireless nodes.

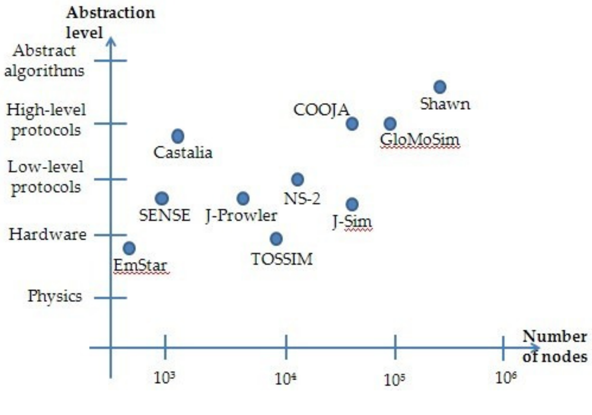


FIGURE 2: Comparison of simulators based on scalability and level of abstraction

Almost all the packages provide graphical support, which makes user can really easily to understand the whole simulation situation. OMNET++ and J-Sim provide powerful GUI libraries for animation, tracing and debugging. Current support in NS-2 is the unelaborated and simple trace reproduction Nam tool. Specific tools also provide surprisingly rich GUIs. TinyViz is the TOSSIM visualization tool, an extensible Java application that provides useful debug information. Besides, it can control and drive the simulation elements. Users can develop their own plugins, which listen for TOSSIM events published by TinyViz and perform some action. However, the latest TOSSIM version 2.x currently does not support TinyViz because the exact radio metadata interfaces have not been determined yet and the code hasn't been written, so if the graphical support is needed, we can use the TOSSIM 1.x instead.

5. CASE STUDY

The aim of this study is to demonstrate a performance comparison for one specific application. It should be however noted that each simulator has a varied range of performance depending on the application and thus the choice of simulator is greatly based on the application. Here, we present a case study to compare the performance of NS2, TOSSIM and Shawn because they are the most popular and very widely used in the WSN area. In this case, a simple application is simulated to broadcast a message every 250ms, the communication range of the sensor nodes is set to 50 length units and each simulation runs for 60 simulated time units. The size of the simulated area is 500x500 length units. A number of simulations with increasing node count were performed. Therefore, the network's density increases steadily as more nodes are added to the scenario. This application has been implemented on TOSIM, NS-2, and Shawn.

Since the exchange of wireless messages is the key ingredient in wireless sensor network, we have presented a measurement to show the required CPU time and memory for each simulator. Figure 3 and Figure 4 shows the time and memory consumption in the three simulators.

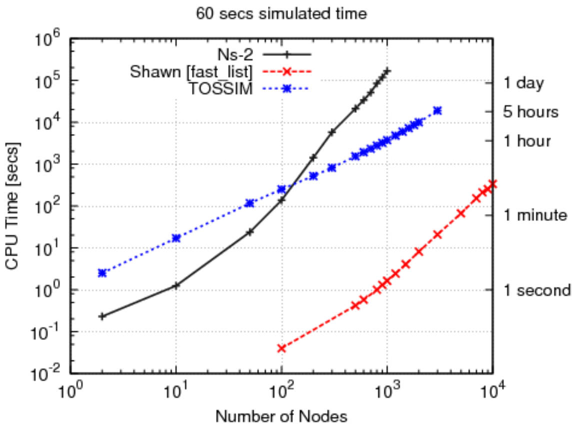


FIGURE 3: Computational time for simulations in NS-2, TOSSIM and Shawn

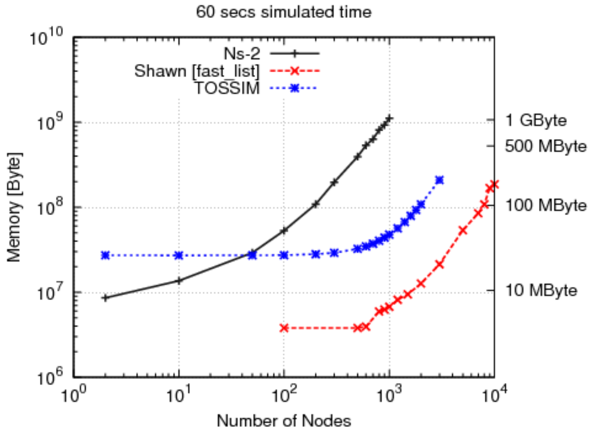


FIGURE 4: Memory consumption for NS-2, TOSSIM and Shawn

Figure 3 shows a comparison of the computation time for different number of nodes in each simulator. From the figure, it can be seen that Shawn can simulate a network with about 10,000 nodes in less than an hour. The performance of NS-2 is good for 100 nodes, which decreases significantly as the number of nodes increase. TOSSIM follows a linear curve. Figure 4 shows the memory consumption for the three simulators. Shawn has the least memory consumption of the three simulators. This clearly shows that Shawn excels in its specialty: the simulation of large-scale sensor networks with a focus on abstract, algorithmic considerations and high-level protocol development.

6. CONCLUSION

The goals of this paper have been to provide comprehensive survey and background on a number of different sensor network simulators and present the pros and cons of each simulator. Knowledge of the strengths and weaknesses of a number of different simulators is valuable because it allows users to select the one most appropriate for their testing. Simulators are compared based on different criteria, and comparative results are presented in tabular form. In addition, short descriptions of simulators are also provided. Since no single simulator under survey is universally applicable to all situations, appropriate guidelines for choosing the best simulator for a particular application environment are also provided.

7. REFERENCES

1. M. Ilyas and I. Mahgoub, Handbook of sensor networks: compact wireless and wired sensing systems, BocaRaton, FL., CRC Press, 2004.

2. J. Liu, et. al., “Simulation modeling of large-scale ad-hoc sensor networks,” European Simulation Interoperability Workshop 2001, London, England, June 2001,

3. I.F. Akyildiz and W. Su and Y. Sankarasubramaniam and E. Cayirci, “A Survey on Sensor Networks,” IEEE Communication Magazine, vol. 40, no. 8, pp. 102-116, Aug. 2002.

4. David Curren, “A survey of simulation in sensor networks,” University of Binghamton, NY, 2005.

5. John Heidemann, Kevin Mills, Sri Kumar, Expanding Confidence in Network Simulations.

6. E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Mariño, J. García-Haro, Simulation Tools for Wireless Sensor Networks

7. Mekni, M. Moulin, A Survey on Sensor Webs Simulation Tools.

8. WeiChung,Hu MingLun, Lee TzungShian, Tsai Hewijin, Christine Jiau, A GUI Simulation Model in Supporting Embedded Software Design
9. David M. Nicol, Scalability of Network Simulators Revisited
10. Ns-2 [Online]. Available: http://www.isi.edu/nsnam/ns/. Retrieved: 02/04/2010.
11. T. Issariyakul and E. Hossain, Introduction to network simulator ns2, Springer, Nov. 2008.
12. P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire tinyos applications," 1st ACM Conference on Embedded Networked Sensor Systems, Los Angeles, CA, Nov. 2003
13. P. Levis and N. Lee, Simulating tinyos networks [Online]. Available: http://www.cs.berkeley.edu/pal/research/tossim.html. Retrieved: 02/04/2010.
14. L. F. Perrone and D. Nicol, "A scalable simulator for TinyOS applications," Proceedings of the Winter Simulation Conference, vol. 1, no. 8, pp. 679-687, Dec. 2002.
15. TinyOS, an open-source operating system for wireless embedded sensor networks [Online]. Available: http://www.tinyos.net/. Retrieved: 02/04/2010.
16. NesC, the sensor network programming language on TinyOS operating system [Online]. Available: http://nescc.sourceforge.net/. Retrieved: 02/04/2010.
17. TOSSIM [Online]. Available: http://docs.tinyos.net/index.php/TOSSIM. Retrieved: 02/04/2010.
18. P. Levis, TOSSIM: Simulating TinyOS Networks [Online]. Available: http://www.eecs.berkeley.edu/~pal/research/tossim.html. Retrieved: 02/04/2010.
19. X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A library for parallel simulation of large-scale wireless networks," SIGSIM Simulation Digest, vol. 28, no. 1, pp. 154-161, 1998.
20. GloMoSim [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/. Retrieved: 02/04/2010.
21. S. Dhurandher, S. Misra, M. Obaidat, and S. Khairwal, "UWSim: A simulator for underwater sensor networks," Simulation, vol. 84, no. 7, pp. 327-338, 2008.
22. J. Cui., J. Kong, M. Gerla and S. Zhou, "Challenges: Building scalable mobile underwater wireless sensor networks for aquatic applications," UCONN CSE Technical Report: UbiNET-TR05-02, University of Connecticut, USA, 2005.
23. R. Jurdak, C. V. Lopes and P. Baldi, "Battery lifetime estimation and optimization for underwater sensor networks," Sensor Network Operations, May 2006. pp. 397–420., Wiley IEEE.
24. Ben L. Titzer, Daniel K. Lee, Jens Palsberg, "Avrora: Scalable sensor network simulation with precise timing," 4th Int. Conf. on Information Processing in Sensor Networks, 2005.
25. Avrora [Online]. Available: http://compilers.cs.ucla.edu/avrora. Retrieved: 02/04/2010.
A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A lightweight and flexible operating system for tiny networked sensors," Proceedings of the 29th Annual IEEE international Conference on Local Computer Networks, Tampa, FL., Nov.2004, pp. 455-462.
26. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," 1st IEEE International Workshop on Practical Issues in Building Sensor Network Applications, pp. 8, Tampa, Florida, USA, 2006.
27. F. Osterlind, "A sensor network simulator for the Contiki OS," Swedish Institute of Computer Science (SICS), Tech. Rep. T2006-05, Feb. 2006.
28. P. J. Marrón, et. al., "COOJA/MSPSim: Interoperability testing for wireless sensor networks," 2nd Int. Conf. on Simulation Tools and Techniques, page 7, Rome, Italy, Mar. 2009.
29. S. Park, A. Savvides, and M. Srivastava, "SensorSim: A simulation framework for sensor networks," 3rd ACM Int. Workshop on Modeling, Analysis & Simulation of Wireless and Mobile Systems, Boston, MA, Aug. 2000.
30. S. Park, A. Savvides, and M. B. Srivastava, "Simulating networks of wireless sensors," Winter Simulation Conference, Arlington, Virginia, Dec. 2001.
A. Boulis, "Castalia, a simulator for wireless sensor networks and body area networks," version 2.0, User's manual, May 2009 [Online]. Available: http://castalia.npc.nicta.com.au/. Retrieved: 02/04/2010.
B. Boulis, "Castalia: Revealing pitfalls in designing distributed algorithms in WSN," 5th Int. Conf. on Embedded Networked Sensor Systems, Sydney, Australia, Nov. 2007.
31. Alexander Kr¨oller, Dennis Pfisterer, S´andor P. Fekete, and Stefan Fischer, Algorithms and Simulation Methods for Topology-Aware Sensor Networks.

A. Kr¨oller, D. Pfisterer, C. Buschmann, S. P. Fekete, S. Fischer, Shawn: A new approach to simulating wireless sensor networks.

32. L. Girod, et al., “EmStar: An environment for developing wireless embedded systems software,” USENIX Technical Conference, Boston, MA, June 2004.
33. L. Girod, et. al., “EmStar: An environment for developing wireless embedded systems software,” Technical report, Center for Embedded Networked Sensing, University of California, Los Angeles, CENS Technical Report 009, 2003.
34. L. Girod, et. al., “Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks,” ACM Transactions on Sensor Networks, vol. 3, no. 3, article 13, Aug. 2007.
35. Familiar Linux [Online]. Available: http://www.handhelds.org. Retrieved: 02/04/2010.
    A. Sobeih and J. C. Hou, “A simulation framework for sensor networks in J-Sim,” Tech. Report UIUCDCS-R-2003- 2386, Dept. of Computer Science, University of Illinois at Urbana-Champaign, November 2003.
36. J-sim [Online]. Available: http://nsr.bioeng.washington.edu/jsim/. Retrieved: 02/04/2010.
37. G. Chen, J. Branch, M. Pflug, L. Zhu, and B. Szymanski, “SENSE: A sensor network simulator,” Dept. of Computer Science, Rensselaer Polytechnic Institute, 2004.
38. SENSE [Online]. Available: http://www.ita.cs.rpi.edu/sense/index.html. Retrieved: 02/04/2010.
39. P. Baldwin, S. Kohli, E. Lee, S. Liu, and Y. Zhao, “VisualSense: Visual modeling for wireless and sensor network systems,” Technical Memorandum UCB/ERL M05/25, University of California, Berkeley, CA, April 2004.
40. VisualSense [Online]. Available: http://ptolemy.eecs.berkeley.edu/visualsense/. Retrieved: 02/04/2010.
41. G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi “Simulation-based optimization of communication protocols for large-scale wireless sensor networks,” IEEE Aerospace Conference, Big Sky, MT, Mar. 2003. Prowler [Online]. Available: http://www.isis.vanderbilt.edu/projects/nest/prowler/index.html. Retrieved: 02/04/2010.
42. JProwler [Online]. Available: http://www.isis.vanderbilt.edu/projects/nest/jprowler/index.html. Retrieved: 02/04/2010.
43. http://www.mathworks.com/products/matlab/

Review on the Design of Web Based SCADA Systems Based on OPC DA Protocol

Hosny A. Abbas hosnyabbas@yahoo.com
Senior automation Engineer
Qena Paper Company
Qus, Qena, Egypt,P.O:83621

Ahmed M. Mohamed ahmed@engr.uconn.edu
Electrical Eng. Department, Aswan Faculty of
Engineering, South Valley University
Aswan, Egypt

Abstract

One of the most familiar SCADA (supervisory control and data acquisition) application protocols now is OPC protocol. This interface is supported by almost all SCADA, visualization, and process control systems. There are many research efforts tried to design and implement an approach to access an OPC DA server through the Internet. To achieve this goal they used diverse of modern IT technologies like XML, Webservices, Java and AJAX. In this paper, we present a complete classification of the different approaches introduced in the litrature. A comparative study is also introduced. Finally we study the feasibilty of the realization of these approaches based on the real time constraints imposed by the nature of the problem.

Keywords: SCADA, OPC DA, Web, IT

1. INTRODUCTION

OPC stands for OLE (Object linking and Embedding) for Process Control - now it also stands for Open Process Control- draws a line between hardware providers and software developers. It provides a mechanism to provide data from a data source and delivers the data to any client application in a standard way. The utility of OPC has now reached the point where automation without OPC is unthinkable. This interface is supported by almost all SCADA, visualization, and process control systems [1]. It gives production and business applications across the manufacturing enterprise access to real-time plant floor information in a consistent manner, making multivendor interoperability and "plug and play" connectivity a reality [2]. Interoperability is assured through the creation and maintenance of open standards specifications. There are currently seven standards specifications completed or in development. Based on fundamental standards and technology of the general computing market, the OPC Foundation adapts and creates specifications that fill industry-specific needs. OPC will continue to create new standards as needs arise and to adapt existing standards to utilize new technology. OPC is based on the DCOM/COM component-object programming model developed by Microsoft in which software is divided into smaller, independent units (the objects). Web-based SCADA system uses the Internet to transfer data between the RTUs (Remote Terminal Units) and the MTU (Master terminal Unit) and/or between the operators' workstations and the MTU. This will reduce the cost of the installation of the SCADA network if compared with installing a dedicated network [3]. Therefore, many researchers all over the world tried to design and to implement an approach to access an OPC DA server through internet to realize a web based SCADA system. DCOM is suitable for LANs where there are less interruptions and noise, but when used through Internet there will be some limitations related to its nature. For this reason, the researchers tried to use IT (Information Technologies) services to achieve their goals. In the following sections, we present an overview of the work in this area.

2. Previous Work

The main challenge will be accessing an OPC DA server, which is a COM/DCOM server through the Internet. We have two options, using DCOM or using modern IT technologies. There are many solution presented in each option. We classify them into the following categories.

2.1 DCOM

Most of the previous research used DCOM for communication with OPC DA server through LANs. For example Xiaofeng Lee et al. [4] uses DCOM communication between an OPC DA client and OPC DA server, then they transfer the OPC DA client data to XML format to be able to access these data through Internet with an XML-DA client which communicates to an XML server (Web server) to get data as shown in Figure 1. Also Truong Chau et al. [5] uses DCOM to enable the C# server script to access OPC DA through LAN and because the OPC DA client is a .NET client, they used an OPC .NET wrapper to make the transformation from .NET to COM and COM to .NET as shown in Figure 2. Zhang Lieping et al. [6] uses the OPC DA Toolbox which is integrated in MATLAB 7 and above editions, this Toolbox enables MATLAB applications to communicate with OPC DA servers through DCOM then the user can simply and conveniently realize the operation to the OPC objects. As they claimed that these features could simplify the process of development and provide an effective method to realize the remote real-time communication between MATLAB and process devices as shown in Figure 3. From the above discussion, we conclude that the only way to communicate directly to the OPC DA server is DCOM (or COM if the client and server are on the same machine). Unfortunately using DCOM through Internet is avoided for many reasons such as, DCOM is windows dependent platform, difficult to configure, has very long and non-configurable timeouts, and cannot be used for Internet communication. That is why DCOM is best suitable for LANs where there are less number of nodes and small delay times. Therefore, the first step to access an OPC DA through web is to use COM or DCOM through LAN, and then we have to do a suitable transformation to enable accessing the OPC data through Internet.

2.2 XML

XML is a platform-independent, which is an important feature to achieve interoperability between different applications, which are running on different platforms. This is the reason, which forced the OPC FOUNDATION to release the OPC XML-DA specifications to allow the XML applications to access OPC data in a standard way. The other advantage of OPC XML-DA is the simple administration as it based on SOAP and XML. On the other hand, it has some disadvantages such as:

- Not suitable for transferring large data volumes
- XML technology is generally slower than COM

- The interaction parameters coded using XML, which leads to an overhead.
- An OPC XML-DA Service is stateless.

During browsing, no information about the position of the client in the namespace is stored in the OPC XML-DA Service, but all information about the namespace (or a defined part of it) transferred to the client at the same time. The client can poll for values at the server, but it should also be possible, to receive changed values automatically. XML-DA servers may stand alone, or may be developed to wrap COM based 3.0, and even 2.0x servers [7] [4].

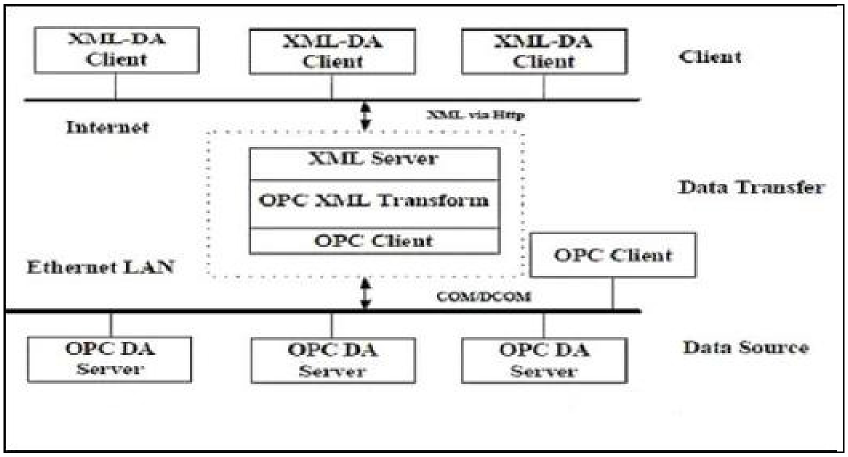


FIGURE 1 : Xiaofeng Lee et al. [4] Design

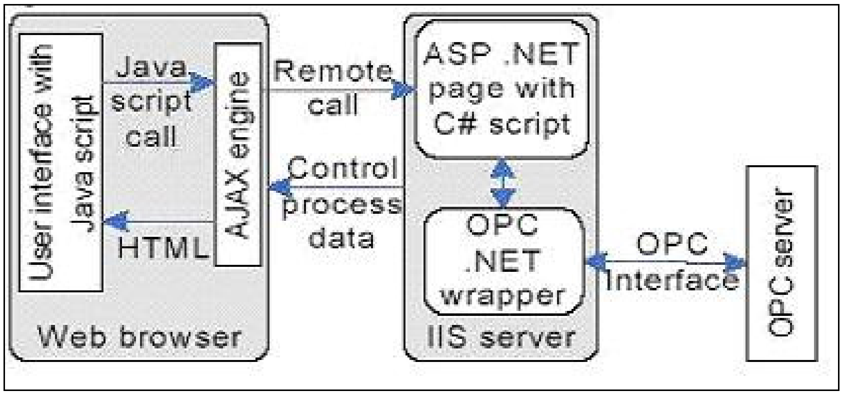


FIGURE 2 : Truong Chau et al. [5] Design

Xiaofeng Lee et al. [4] suggested designing an information integration system which will adopt OPC DA to OPC XML-DA; the design includes three layers structure, data source layer, data transfer layer and client layer as shown in Figure 1. The authors claim that because of adopting industry standard OPC interface and webservice transfer interface, the remote monitoring system based on OPC XML-DA technology makes it convenient to update and expand system. If we analyze this approach, we will find that there is an overhead in layer2 because of the COM-XML transformer. In addition, the authors did not expose to the problem of client data update, is there a data polling mechanism that enables the client to get the new data- if there is –in an efficient way that consumes as little as possible of available resources. For similar work, that uses XML and/or OPC XML-DA techniques see [8, 9, 10]. Due to possible performance limitations, OPC XML-

DA is unlikely used for real time applications, although it is commonly used as a bridge between the enterprise and control network. Furthermore, only OPC-DA functionality is provided in XML-DA. So it can be best seen as a transitional path to a true Webservice architecture that is just released by the OPC Foundation, which is OPC-UA (Unified Architecture) project [11].

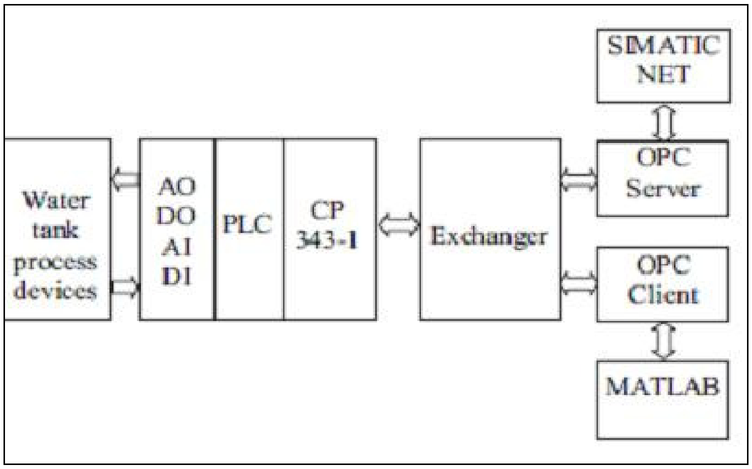


FIGURE 3: Zhang Lieping et al. [6] Design

2.3 Webservices

Webservices as defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network. As Shekhar M. Kelapure et al. [12] mentioned about the Webservices features, which are:

- Communicate via open protocols (HTTP, SMTP, etc.)
- Processes XML messages framed using SOAP
- Describes its messages using XML Schema
- Provides an endpoint description using WSDL
- Can be discovered using UDDI

The authors found that they could achieve a good web based SCADA system using Webservices as shown in Figure 4. They concluded that the advantages obtained when using Webservices in SCADA systems are:

- Web services provide data publishing on the internet through HTTP protocol, thereby eliminating the need to compromise on security of SCADA servers.
- Webservices provide the means to publish the data through various devices like thin clients, PDAs and mobile phones.
- Webservices facilitate interfacing of multiple control centers, like ICCP, once the services are standardized.
- Webservices are supported by .NET as well as JAVA technologies.
- Data can be fetched from Hard Disc as well as RAM using Web-services

On the other hand, Webservices have some disadvantages like:

- Webservices standards for features such as transactions are currently nonexistent or still in their infancy compared to more mature distributed computing open standards such as CORBA.

- Webservices may suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off when choosing text-based formats. XML explicitly does not count among its design goals either conciseness of encoding or efficiency of parsing.
- In addition, as mentioned in [13] that by utilizing HTTP, webservices can evade existing firewall security measures whose roles are intended to block or audit communication between programs on either side of the firewall.

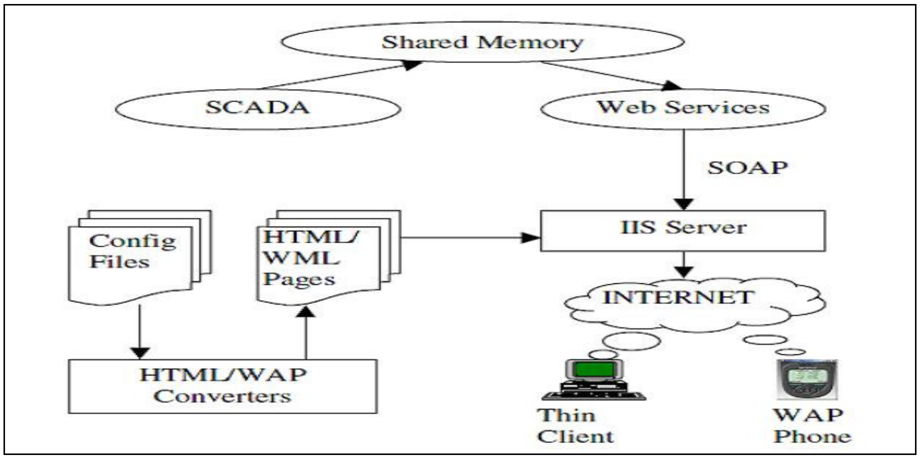


FIGURE 4: Shekhar M. Kelapure et al. [12] design

Nunzio M. Torrisi et al. [14] proposed what they called CyberOPC which is a communication system anticipates the use of a gateway station called CyberOPC gateway, which will process messages sent to the OPC towards the public network and vice versa. Their proposed communication system targeted to best effort network with minimum bandwidth reserved for periodic traffic as shown in Figure 5.

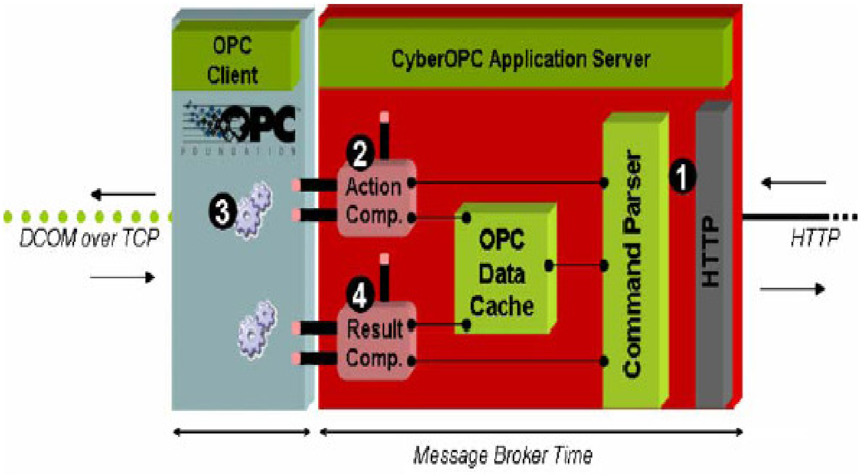


FIGURE 5 : Nunzio M. Torrisi et al. [14] Design

As they claimed that the necessity to satisfy time-critical and security requirements for remote control has stimulated the study of a new protocol for process control. Moreover, to obtain maximum interoperability with existing factory floor technologies, they built their communication project over the OPC technology. Unlike many OPC gateways for the Internet that use Webservices with HTTP and SOAP, the CyberOPC gateway does not use Webservices because the loss in performance will not be balanced by the advantage of a high-level programming language offered by Webservices. However, in order to use Webservices that transport OPC data, OPC libraries for processing OPC messages are required. Therefore, assuming that the use of libraries is necessary for processing non-open source and usually not free OPC data, they believe that it is useful to develop a set of free and open source libraries in order to implement the OPC communication over the Internet with the best possible performance. This approach does not solve the problem of periodic data update of the remote client to get new data. Moreover, when the remote client makes a request, the CyberOPC will not check if there is a change in this data from the last sent data or not, so if the control process has a high frequency data change rate, the remote client has to increase its periodic data-requesting rate, which can affect the server efficiency and network bandwidth.

2.4 Classic Approaches

Thomas Bauer et al. [15] claimed that to integrate OPC technology with Internet, its necessary to have an asynchronous data transmission method that is time-independent bidirectional communication, which is not possible because of the architecture of Internet. Therefore, they suggested an approach to solve this problem as shown in Figure 6. As they mentioned, first the Client (browser) will initiate a connection request for establishing at least one transmission channel that should be permanently open to send data at any desired time (asynchronously) and independently of the action of the user. In order to keep this data connection permanently open the web server should continuously send data to the client, if there is no useful data it will send dummy data or send information to tell the client that a useful data coming. The need of dummy data is to maintain the data connection. Keeping this channel permanently open will enable the server to initiate the communication to the client, in the same time the client is still can initiate a new request to the server using a another channel. In other words, the permanent channel will be for the server and the other channels are for the client. The main disadvantage of this approach is that the server and network will suffer because of the continuous transmit of useful and unuseful data. What will happen if the data change rate is slow? The web server will send a dummy data for long time wasting the network bandwidth with no feasibility. Also what will happen if for some reason the permanently open channel broken? The client will have to reinitiate the channel again. In addition to that, the client must keep checking the permanent open channel for new useful data the action that can affect the user interactivity.

Duo Li et al. [3] suggested an approach to implement the function of real time monitoring which provides periodically updated data to operators. The target data, which saved in a database, is marked and mapped to an HTML file-, say file2- in a web server; the database server automatically refreshes file2 with the latest data whenever the target data in the server updated. To show target data in an HTML file a Java applet developed that connects to file2 to get the latest data periodically and display them in the required format, another HTML file -say file1- in which the applet is included, developed to establish a basic human-machine interface (HMI) and complete some initiations. Data monitor functions commence from a web browser sending an HTTP GET command to the web server asking for file1, which then fetched to the browser and displayed and the applet is downloaded to run to show the target data as shown in Figure 7.

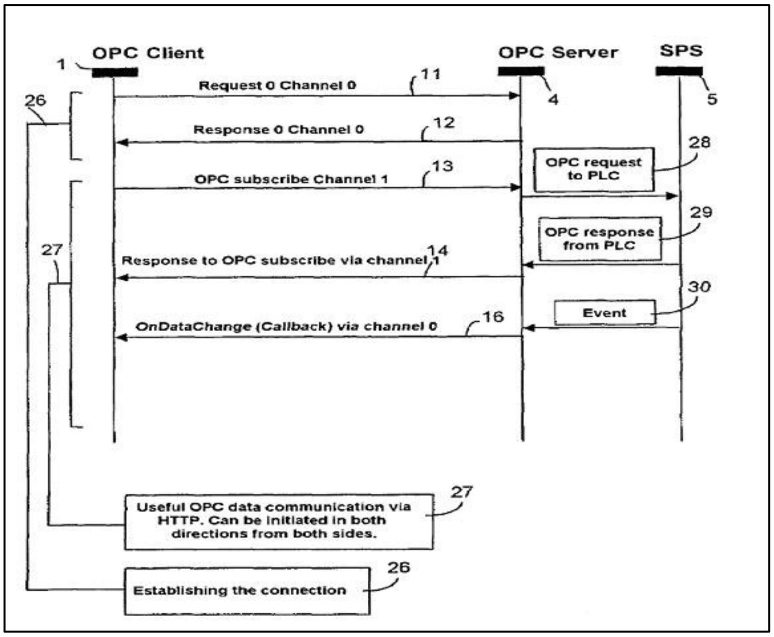

OPC Client
OPC Server
SPS
1
4
5
Request 0 Channel 0
11
Response 0 Channel 0
12
OPC subscribe Channel 1
13
Response to OPC subscribe via channel 1
14
OnDataChange (Callback) via channel 0
16
OPC request to PLC
28
OPC response from PLC
29
Event
30
26
27
Useful OPC data communication via HTTP. Can be initiated in both directions from both sides.
27
Establishing the connection
26

FIGURE 6 : Thomas Bauer et al. [15]

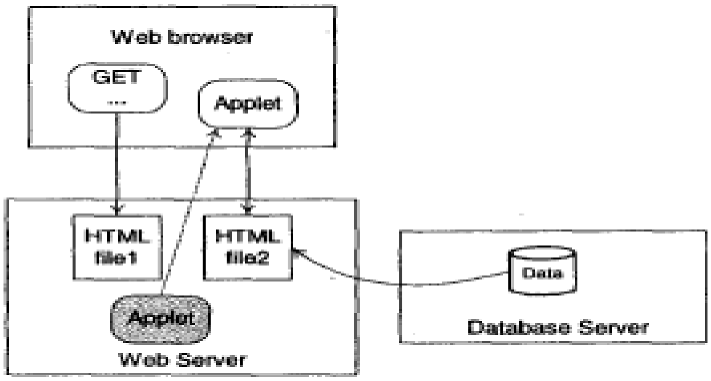


FIGURE 7 : Duo Li et al. [3] Approach

Actually, Java is a perfect programming language especially for web application development because its programs after compiled transferred to what known as byte code format, which can executed by Java virtual machine (JVM) that is now supported by most platforms so a good interoperability can be achieved by Java. In addition, Java has many other features like Java applets and Servlets. Applets run on client (browser) and Servlets run on a web server. Applets and Servlets can communicate with each other in an efficient and persistent way, so they used to develop web based SCADA systems. On the other hand, they have some disadvantages:

1. Java plug-in is required to run applet
2. Java applet requires JVM so first time it takes significant startup time.
3. If the applet not already cached in the machine, it downloaded from Internet and will take time.
4. Its difficult to design and build good user interface in applets compared to HTML technology

Disadvantages of Servlets

1. Developers MUST know JAVA.
2. Web Administrator will need to learn how to install and maintain Java Servlets
3. Tedious uses of out.println() statements
4. Can be remedied by using Java Server Page (JSP)

Mike Clayton et al. [16] designed a TCP client-server Java architecture based on socket support to develop applications to integrate SCADA systems and Web applications. As he mentioned, that he achieved a solution to have a bridge between two complex applications (SCADA software and web servers), which evolve independently because they belong to different worlds. The proposed design is shown in Figure 8, which illustrates the TCP socket communication between the web server and the SCADA system. As shown in the figure that there is a Java application on the SCADA system machine, this Java application is a server, which allows the TCP connections from the remote clients. The Java application or the server communicate with the SCADA system by JNI (Java native interface) which enables Java classes to communicate with standard programs using custom librarries in C/C++. For each client TCP connection request from the web server, the Java server will create a new thread to handle that connection.

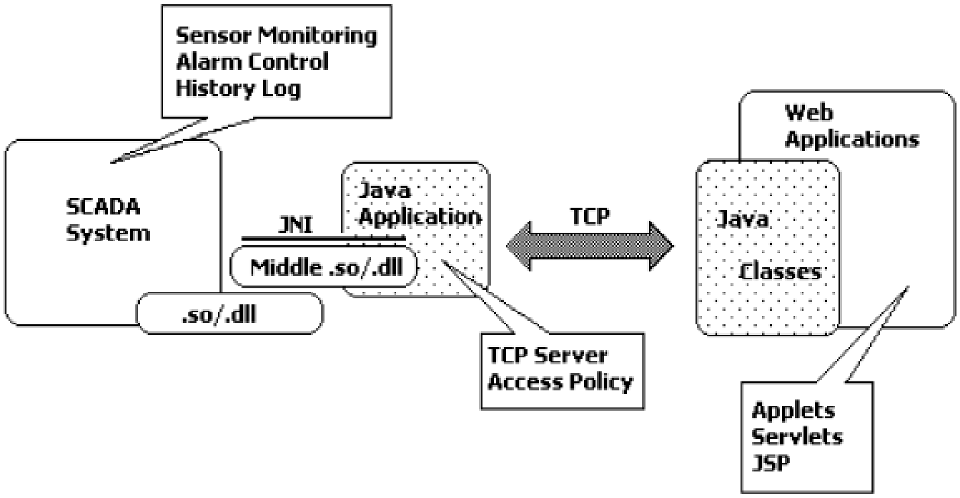


FIGURE 8 : Mike Clayton et al. [16] Approach

This approach shares the same disadvanteges like Duo Li et al. [3] approach, in addition to the complexity and non-flexibility in the Java application (server), because if the SCADA system changed or updated the Java application will need to be modified to use a new custom library. Also with large number of clients, the SCADA machine (Java server machine) will be very loaded with threads that may impacts the SCADA system performance. In addition, the author of this approach concentrated on the communication between the web server and the JAVA server on the SCADA machine and ignored the client to the web server (browser), how the browser updates its data with new data. There should be a data polling mechanism, which enable the browser to ask for data update.

S.H. YANG et al. [17] claimed in their guidance for web based Process control systems that Internet can be linked with the local computer system at any level in the information architecture, or even at the sensor/actuator level. These links result in a range of 4Rs (response time, resolution, reliability, and reparability). For example, if a fast response time is required a link to the control loop level should be made. If only abstracted information needed, the Internet should linked with a higher level in the information architecture such as the management level or the optimization level as shown in Figure

9. In addition, they mentioned that the Internet is a public transmission media, which is fundamentally different from other private transmission media used by many end-users for different purposes. The Internet transmission performance is associated with time delay and packet loss and possesses large temporal and spatial variation. In detail, the Internet time delay is characterized by the processing speed of nodes, the load of nodes, the connection bandwidth, the amount of data the transmission speed, etc. Therefore, it is somewhat unreasonable to model the Internet time delay for accurate prediction at every instant. In addition, they exposed to the problem of concurrent users, as they said that the special feature of the Internet-based SCADA is multiple-users and the uncertainty about users. The number and location of the users keep changing and the operators cannot see each other or may never have met. It is likely that multiple-users may try to control concurrently a particular process variable in which case some problems may arise. So, coordination among multiple-users becomes very important.

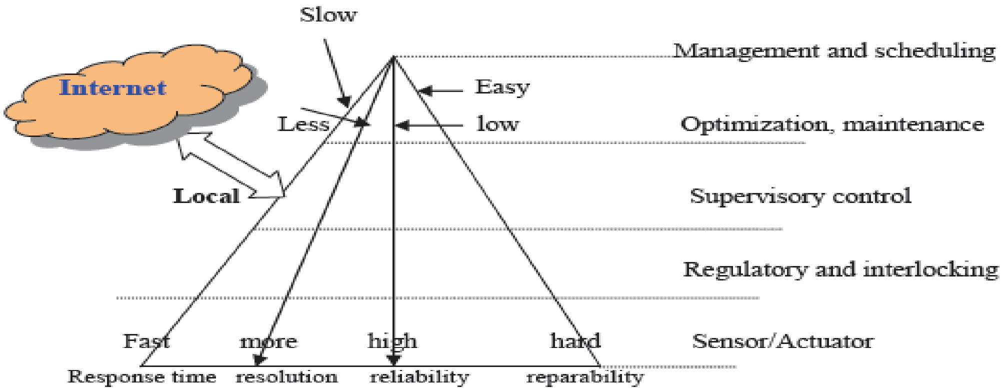


FIGURE 9 : S.H. YANG et al. [17] Guidance (information architecture levels)

2.5 AJAX

Actually, AJAX (Asynchronous JavaScript and XML) started a new line in web based SCADA systems because it enables us to create more interactive web pages, which are our way to replace traditional desktop SCADA application. AJAX considered as the future of the Internet because of its ability to simulate desktop applications by the new features it offers like asynchronous client-server calls and partial-page updates. Ajax is a group of interrelated web development techniques used on the client-side to create interactive web applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. The use of Ajax techniques has led to an increase in interactive or dynamic interfaces on web pages. Truong Chau et al. [5] used AJAX in their approach to get the OPC DA data to the client web page with high user interactivity as shown in Figure 2. But because of the problems of the HTTP protocol as a stateless protocol they had to use a simple data polling mechanism by using an AJAX Timer to poll for new OPC DA data set from the OPC DA server through the web server. When using an AJAX style of programming the old, classic programming approach must be given up. There is no form submit any more that posts all the client state to the server and requests for a complete new page description using HTML. Instead of loading several pages until the functionality done, only one page loaded and stay in the browser until the end of functionality. With the Ajax model, the execution of the web page processed in three different phases: loading the page, loading data from the server using AJAX techniques and interaction with the page using AJAX techniques

The web standards used in AJAX are well defined, and supported by all major browsers. AJAX Applications are browser and platform independent. The main disadvantage of AJAX is security. Sometimes developers do not put checks on the data coming into the server - they assume that it is coming from their own website. Unfortunately, this is subject to injection attacks. Furthermore, there are several ways to "fake" how the data is coming in and making detection to all is impossible. Another disadvantage is that AJAX does not play well in encrypted environments. AJAX relies on plain text transmission (nothing but text can be transmitted through AJAX anyways), and so encrypting this stream and having the server-side program deals with it presents large problems [18].

3. COMPARATIVE EVALUATION

None of the mentioned approaches considered the feasibility of the realization of their work. If we look closely to [4, 8, 9, 10] we will find that they use XML and/or OPC XML-DA technologies, which have the disadvantages such as:

- Not suitable for transferring large data volumes
- XML technology is generally slower than COM
- The interaction parameters coded using XML, which leads to an overhead.
- An OPC XML-DA Service is stateless.

Actually, OPC XML-DA designed for Internet access and enterprise integration and based on its platform-independence; it mainly implemented in embedded systems and on non-Microsoft platforms. However, due to its high resource consumption and limited performance, it was not as successful as expected for this type of applications.

In [12] the authors used the features offered by Webservices to design and implement a web based SCADA system. Moreover, they could solve the challenges, which they faced such as thin client support, refresh of GUI window, and firewalls restrictions. However, they could not solve real time data collection challenge in an efficient and effective way because they used a constant refresh frequency of a few seconds, which will lead to non-synchronized process data transfer.

In [15], to allow each node (client or server) to start sending data to the other node independently at any time, the authors had to maintain a connection in a permanent open state by making the server continuously sends useful or unuseful data to the client. In addition, this will affect the performance of the system especially with large number of clients and low network bandwidth, and this may lead to server crash.

In [3] the authors will need to find a way to periodically update file2, which may be a loop or any other way. Also the java applet, which embedded in the HTML file1, will need to use a Timer periodically to get the new data from file2 and then they will face the problem of synchronization between the applet Timer interval and the process of updating file2 with new data and this will be difficult. In addition, they ignored the heavy network load and the need for large server memory to handle the large number of requests and the heavy load on the server CPU. It is very difficult (nearly impossible) to guarantee a real time behavior with such limitations. The only solution is to spend more money to increase the network bandwidth and the server CPU speed and memory capacity (The hardware resources).

In [14], the designed CyberOPC does not solve the problem of periodic data update of client to get new data. Moreover, when the remote client makes a request, it will not check if there is a change in these data from the last sent data or not so, if the control process has a high frequency data change, the remote client has to increase its periodic data requests, which affects the server efficiency and network bandwidth.

In [5], the authors did not take any attention to the efficiency of their approach. The Timer, which they used, will get the data without any care if there is a significant data change or not. In addition, how they can specify the Timer interval, there are two cases for that, first, when the interval is small i.e. one second (high frequency data polling), where the server and network will suffer. Second, if the interval is large they will loss some data change events that maybe very important and the system will not considered as a Real-Time monitoring system. Therefore, this approach still needs some modifications, which enable us to get better efficiency and real time behavior.

4. CONSLUSION & FUTURE WORK

None of the mentioned approaches considered the feasibility of the realization of their work. No one of them succeeded to achieve a real time behavior, which is very important in the functionality of the SCADA systems, and especially web based SCADA systems. In addition, no approach addresses the efficiency and effectiveness of the designed system. They did not pay any attention to the consumption of the system available resources such as CPU load (of the web server) and Network bandwidth. These issues are very important to get a more reliable and available SCADA system. In future, all the researchers in this area should take care of these points to make web based SCADA applications competitive to traditional desktop applications.

5. REFERENCES

1. OPC Foundation, "*OPC DA 3.0 Specification [DB/OL]*", Mar.4, 2003
2. http://www.controlglobal.com/articles/2004/229.html
3. Duo Li, Yoshoizumi Serizawa, "*Concept Design for a Web-based SCADA system*", Proceedings of Transmission and Distribution Conference and Exhibition 2002: Asia Pacific, IEEE/PES, pp:32 - 36 vol.1 2002 JAPAN
4. Xiaofeng Lee 1, Jianfeng Hu1+,"*Design and Research of Remote Monitoring System based on OPC XML-DA*", Proceedings of 2008 International Pre-Olympic Congress on computer science, V(1), pp:147-151, August 2008, China
5. Truong Dinh Chau, Nguyen Ngoc Khai, "*WEB-BASED DATA MONITORING AND SUPERVISORY CONTROL*", Proceedings of the International Symposium on Electrical & Electronics Engineering. Oct. 2007.
6. Zhang Lieping, Zeng Aiqun, Zhang Yunsheng, "*On Remote Real-time Communication between MATLAB and PLC Based on OPC Technology*" Proceedings of the 26th Chinese Control Conference July 26-31, 2007, China
7. OPC Foundation, "*OPC XML-DA Specification Version 1.0*", Released July 12, 2003.
8. Vu Van Tan, Dae-Seung Yoo, and Myeong-Jae Yi, "*Design and Implementation of Web Service by Using OPC XML-DA and OPC Complex Data for Automation and Control Systems*", Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT'06) 2006, Korea

9. Shamdutt Kamble, Venkateswara Rao M, Shailendra Jain, "*OPC Connectivity to Remote Monitoring & Control*", White paper, Wipro Technologies company, www.wipro.com, 2004

10. Thomas Dreyer, David Leal, Andrea Schröder, and Michael Schwan," *ScadaOnWeb – Web Based Supervisory Control and Data Acquisition*", Proceedings of 2nd international semantic web conference, pp. 788-801, FL, USA 2003.

11. Byres research, OPC security WP#1, July 27, 2007

12. Shekhar M. Kelapure_ S.S.K. Sastry Akella† J. Gopala Rao, "*Application of Web Services In SCADA Systems*", The International Journal of Emerging Electric Power Systems V. 6, Issue 1, 2006.

13. http://www.w3.org/TR/WD-script-970314

14. Nunzio M. Torrisi & João F. G. Oliveira," *Remote control of CNC machines using the CyberOPC communication system over public networks", The International Journal of Advanced Manufacturing Technology, V.39, pp. 570-577, 2007*

15. Thomas Bauer, Roland Heymann, Heinz-Christoph, "*System and method for transmitting OPC data via Internet using an asynchronous data connection*", US patent number: US 7,302,485 B2, 2007.

16. Mike Clayton EP - ESS, Philippe Gras EP – HC, Juan A. Oses EP – ESS, "*A SCADA-WEB INTERCONNECTION WITH TCP IN JAVA*", September 24, 2002, *URL: http://ess.web.cern.ch/ESS/GIFProject/PVSSJava/pvssweb.0.8.pdf*

17. S.H. YANG, L.YANG,"*Guidance on Design of Internet-based Process Control Systems*", ACTA AUTOMATICA SINICA, V.31 NO.1 Jan 2005.

18. http://www.helium.com/items/580436-advantages-and-disadvantages-of-ajax

DRSTP: A Simple Technique for Preventing Count-to-Infinity in RSTP Controlled Switched Ethernet Networks

Syed Muhammad Atif syed.muhammad.atif@gmail.com
M.S Computer Networks
Department of Computer System Engineering
Usman Institute of Technology
Karachi, Pakistan.

Abstract

Ethernet is a dominant local area network (LAN) technology from last three decades. Today most LANs are switched Ethernet networks. Spanning tree protocol is a vital protocol for smooth operation of switched Ethernet networks. However the current standard of spanning tree protocol for Ethernet – commonly known as Rapid Spanning Tree Protocol or in short RSTP – is highly susceptible to classical count-to-infinity problem. This problem adversely effects the network convergence time, depending upon how long count-to-infinity situation persists in the network, and thus leads to network congestion and packet loss. In the worst case, even forwarding loops may be induced that further enhances the network congestion. Thus, the dependability of RSTP controlled Ethernet networks are highly questionable due to its vulnerability against the count-to-infinity problem. This paper first discusses the count-to-infinity problem in spanning tree controlled Ethernet networks, in general and in RSTP controlled Ethernet networks, in particular. Then this paper proposes a simple solution to overwhelm this problem efficiently.

Keywords: Network Reliability, Count-to-Infinity, Network Convergence, RSTP.

1. INTRODUCTION

For last three decades, Ethernet is the most prominent local area network (LAN) technology. It can be seen everywhere from home offices to small offices and from medium size companies to even in large enterprises. Ethernet is usually preferred over its contemporary technologies – such as Fiber Distributed Data Interface (FDDI), Copper Distributed Data Interface (CDDI), Token Ring and Asynchronous Transfer Mode (ATM) – because of its low cost, market availability and scalability to higher bandwidths. Today, there are millions of Ethernet stations world-wide and large numbers of applications are running on them. Due to this ubiquity of Ethernet, and the ever-decreasing cost of the hardware needed to support it, most manufacturers now build the functionality of an Ethernet card directly into PC motherboards, obviating the need for installation of a separate network card.

Ethernet was originally developed at Xerox PARC in 1973. In its most basic form, Ethernet is a shared medium in which stations are not explicitly required to know location of each other. This scheme works well when the numbers of stations are few tens. As number of stations on the medium increases the performance and throughput of Ethernet decreases dramatically. To cope with this problem, Ethernet switches were introduced. Ethernet switch is a multi-port network device that forwards frame to specific ports rather than, as in conventional hub, broadcasting every frame to every port. In this way, the connections between ports deliver the full bandwidth available. Since these Ethernet switches works transparently, thus other network devices are completely unaware of their presence. With the introduction of switch, the performance, throughput and scalability of Ethernet has been significantly improved. Today most Ethernet networks are point-to-point switch networks. This study focuses on the dependability of such Ethernet networks under partial network failure.

Every Ethernet switch maintains a table – usually called forwarding table – in a local cache to forward incoming frames. Every entry in the forwarding table has a MAC address and the associated switch port. For each incoming frame, the switch looks up its destination address in the forwarding table to find the switch port to which the address is associated. If it is found, the frame is forwarded out that switch port. Otherwise the frame is forwarded in a “best effort” fashion by flooding it out all switch port except the one that received it. This is known as unknown unicast flooding. Further switches use dynamic address learning mechanism to populate forwarding table. When a frame **F** with a source address **S** arrives at switch port **P**, the switch create an entry in the forwarding table by assuming that the same port **P** can also be used to forward frame destined to **S**. Support of unknown unicast flooding and dynamic address learning by the switches impose a requirement that the underlying network must be cycle free. The reasons for this are two-fold. First is to avoid broadcast and unknown unicast frame from circulating forever in the network. Because, unlike IP, Ethernets frame do not have a Time-to-Live (TTL) field. Second is to prevent address learning mechanism from malfunctioning. Because a switch may receive frames from a station via multiple switch ports in cyclic network.

Ethernet networks usually have redundant links to increase network availability. However, networks that have redundant links also contain cycles and thus violating the requirement needed for proper functioning of switches. To alleviate this problem, switches in the network distributedly computes an active tree topology – by definition, tree has no cycle – over the underlying network that spans all the switches in the network to maintain full network connectivity. Each switch in the network places some of its ports in active mode while other in standby mode. Set of ports in the active modes form a spanning tree and only those are used for forwarding frames. Whereas ports in standby mode are reserve for use in case of link or switch failure. Protocols used by switches to compute such a tree topology are called spanning tree protocols.

The dependability of Ethernet therefore heavily relies on the spanning tree protocol under use. However, there are some serious concerns about the reliability of the current Ethernet standard spanning tree protocol – commonly known as Rapid Spanning Tree Protocol [1]. Cisco documented some pathological causes for forwarding loops in RSTP [2]. It also provides some proprietary solutions such as Loop Guard [3] and Unidirectional Link Detection (UDLD) protocol [4] to address few specific problems of RSTP. Elmeleegy et al. shows that count-to-infinity problem may be exhibited by RSTP under certain conditions [5]. They also proposed Etherfuse [6], a device to alleviate the problem of count-to-infinity in the existing network The incident of network disruption at the Beth Israel Deaconess Medical Center in Boston – in which the network suffered from disruptions for more than three days due to problems with the spanning tree protocol – also proved that the concerns about the reliability of RSTP are quite genuine. Vulnerability of RSTP against count-to-infinity problem is the main cause of its unreliability. This paper discuses count-to-infinity problem in spanning tree controlled network, in general, and in RSTP controlled network, in particular. The paper also provides a simple yet effective solution to prevent this problem by extending RSTP.

The rest of paper is organized as follows. Section 2 gives a brief overview of RSTP. Section 3 explains the conditions under which a spanning tree controlled network may suffer from count-to-infinity. Section 4 elaborates count-to-infinity problem in RSTP. Section 5 describes my proposed solution to this problem, the Delay Rapid Spanning Tree Protocol (DRSTP). Section 6 discusses related work. Followed by, Section 7 which concludes the paper.

2. OVERVIEW OF SPANNING TREE PROTOCOLS

Spanning Tree Protocol (STP) is the earlier standard Ethernet spanning tree protocol. It was first proposed by Perlman in [7]. Current standard Ethernet spanning tree protocol – known as Rapid Spanning tree Protocol (RSTP) [1] – is a modified version of STP. It inherits all the basic concepts of STP but design in such a manner that it is much faster than STP. RSTP [1] is here mainly due to the work of Mick Seaman presented in [8], [9], [10], and [11]. This section gives a brief overview of both these protocols. STP requires a unique identifier (ID) for every switch and every port within a switch. Using a distributed procedure, it elects the switch with the smallest ID as the root. A spanning tree is then constructed, based on the shortest route (path) from each switch to the root (switch and port IDs are used to break ties). The routing information is exchanged in the form of Bridge Protocol Data Units (BPDUs). The port that has received the best information for a route (path) to the root is called the root port. Other ports in the switch send BPDUs with their path cost to the root to other switches in the network. Ports that receive inferior information than the one they are sending are chosen to be designated ports. A port is said to be backup port if it receives superior information transmitted by its own switch. All remaining ports are alternate ports. Every switch brings its root port and its designated ports into a forwarding state thus only these ports are used to forward data frames. All remaining ports – alternate and backup ports – are kept in a blocking state and thus are not used for data forwarding.

In the event of a topology change, STP depends upon timers before switching ports to the forwarding state. This is to provide enough time for the new information to spread across the network. These conservative timers are used to guard against prematurely switching a port to the forwarding state that may lead to a forwarding loop. Due to these timers convergence time of STP may be up to 50 seconds [2]. Whenever a switch gets disconnected from the Root Switch, it waits until the information cached at its root port is aged out, then it starts accepting BPDUs from other switches to discover another path to the root.

In STP, only Root Switch generates BPDU. All non-root switches wait to receive them on their root ports then relay to their designated ports after adjusting the appropriate fields such as Root Path Cost, Sender Bridge Identifier etc. A switch losing a BPDU can be due to a problem anywhere along the path to the Root Switch.

RSTP [1] preserves all the basic concepts of STP but introduce few optimizations to reduce convergence time. Those are,

1. RSTP switches can process inferior BPDUs to detect topology changes.
2. When an RSTP switch is connected to point-to-point links, it uses handshake (sync), rather than timer to transition a Designated Port to forwarding state.
3. If the Root Port of a switch fails, RSTP can quickly retire the Port and make an Alternate Port its new Root Port. This new Root Port can be placed in the forwarding state without any delay.
4. In RSTP, every switch sends its own BPDUs whether it received one on its Root Port or not. RSTP switch expects to receive a BPDU within three Hello times. If the BPDU is not received within this time, the switch presumes it had lost connection with its neighbor. Of course, if a switch detects a loss of a link on its own port, it immediately assumes its neighboring connection is lost.

A topology change can result in the invalidation of a switch's learned address location information. This requires the flushing of the forwarding table that caches stations' locations. Both STP and RSTP [1] use some sort of address flushing mechanism. But address flushing mechanism of RSTP [1] is much faster than that of STP.

3. COUNT-TO-INFINITY IN SPANNING TREE CONTROLED NETWORKS

Count-to-infinity problem is not new to the world of routing. All known distance vector routing protocols such as RIP [12] and EIRP [13] employ some sort of mechanism to encounter this problem. However, this problem is still new to the world of switching. It was first mention by Mayer at el. [14] in 2004 that RSTP [1], a well known spanning tree protocol, may exhibit count-to-infinity problem under certain conditions. This highly undesirable behavior of RSTP was later studied in detail by Elmeleegy at el. [5]. This section will explain why and when a spanning tree protocol may become vulnerable to count-to-infinity.

In a fully converged spanning tree controlled network all alternate ports are dual rooted i.e. have two distinct path to the Root Switch. One path of an alternate port to the Root Switch passes through its link's designated port while the other path passes through its switch's root port. However an alternate port may loss its one or both paths to the Root Switch if the root port of its upstream switch fails. So in a network in which a switch suffering from the root port failure, an alternate port may have no, one or two path(s) to the Root Switch and thus will be called orphan, single rooted and dual rooted alternate port respectively in this text. Orphan alternate ports must not be used to reunite a network segregated due to the root port failure of a switch. Because such alternate ports have information which is no longer valid. Moreover, dual rooted alternate ports are not used by spanning tree protocols to prevent forwarding loops. This left only single rooted alternate ports that can be used to reunite the temporarily segregated network and they have the potential to do so. Hence the underlying spanning tree protocol must use only single rooted alternate ports to restore connectivity.

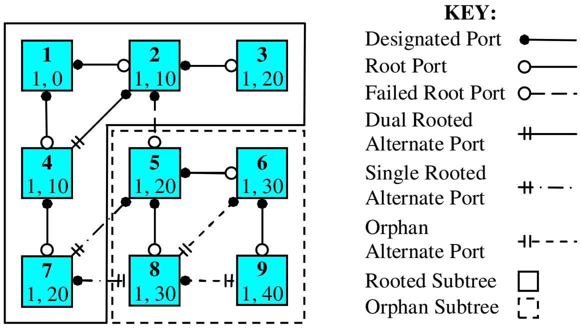


FIGURE 1: Different types of alternate ports in a network after failure of the root port of switch 5.

In a fully converged spanning tree controlled network, failure of the root port (or the designated port associated with the root port) of a switch results into segregation of underlying spanning tree into two distinct subtrees namely a *rooted subtree*, a subtree that still have the Root Switch, and an *orphan subtree*, a subtree that no longer have the previous Root Switch. It has to be noted that since all the switches in the *orphan subtree* have lost their path to previous Root Switch through their respective root ports. Therefore dual rooted alternate ports cannot exist in *orphan subtree*. In contrast, all the switches in *rooted subtree* have a path to the Root Switch through their respective root port. Hence orphan alternate ports cannot exist in *rooted subtree*. However, single rooted alternate ports can be found in both subtrees near their common boarder. An alternate port in the *rooted subtree* is single rooted if and only if its associated designated port is in the orphan sub tree. Similarly an alternate port in the *orphan subtree* is single rooted if and only if its associated designated port is in the *rooted subtree*. These facts are depicted in Figure 1 through an exemplary network. Each switch is represented by a small box. The top number in the box is the Switch ID, the lower set of numbers represents the Root Switch ID as perceived by the switch and the cost to this Root Switch. All links have cost of 10. Figure 1 shows the snapshot of network immediately after failure of the root port of switch **5**. Switches **1** to **4** and switch **7** are in *rooted subtree* and switch **5**, **6**, **8** and **9** are in *orphan subtree*. Alternate port of switch **4** is still dual rooted as it is inside the *rooted subtree*. Moreover, Alternate port of switch **7** and that of switch **8** connected to switch **7** are single rooted alternate ports as they are near the common boarder of two subtrees. While alternate of switch **8** connected to switch **6** and that of switch **9** are orphan alternate ports as they are inside the *orphan subtree*.

Switches in a spanning tree controlled network use messages to communicate with each other. These messages experience a transmission delay when passing through the network. Thus, failure of the root port of a switch may put all its downstream switches, that is switches in *orphan subtree*, into an inconsistent state for a period of time. The absolute period of inconsistence for a switch **B** is from the time when one of its upstream switch's root port (or the designated port associated with the upstream switch's root port) fails to the time when this information will be received on the root port and all alternate ports (if any) of the switch **B**. The effective period of inconsistence for a switch **B** is a bit small and it spans from the time when the first time switch **B** receives failure information of its upstream switch's root port on its root (or alternate) port to the time this will be received on all its remaining alternate port(s) (and the root port). Clearly, only inconsistent switches may have orphan alternate port(s) because of lack of information. Further, such switches cannot differentiate an orphan alternate port from the other two types of alternate ports.

Count-to-infinity only occurs in the part of network constituting the *orphan subtree*, if six conditions are satisfied simultaneously. Three of them have to be satisfied by an inconsistent switch **B**:

1. Switch **B** has an orphan alternate port **a** such that its root path cost is smaller than that of the best single rooted alternate port in the network.
2. Switch **B** starts to declare its orphan alternate port **a** as designated port or the root port when it is still in the effective inconsistent port or switch **B** is declaring its orphan alternate port **a** as designated port when it is entering into the absolute inconsistent state.
3. Switch **B** is injecting the stale BPDU through its retiring orphan alternate port **a** that is becoming designated port or through its retiring root port that is becoming the designated port because the orphan alternate **a** is becoming the new root port.

Two conditions must be satisfied by an upstream switch **A** along with above three conditions:

4. Switch **A** accepts the stale BPDU, transmitted by switch **B**, on its designated port **d**, as it is conceived as superior BPDU by switch **A**. This makes port **d** the new root port of switch **A**. It may happen only if the switch cannot differentiate between stale and fresh BPDUs.
5. Switch **A** begins to propagate the stale BPDU further through its now designated ports.

One condition needs to be met by underlying network.

6. There is at least one (unbroken) cycle in the network passing through switch **A**'s new root port **d** and switch **B**'s orphan alternate port a.

The first and the last condition for count-to-infinity are unavoidable in a high available fault tolerant network. However, remaining conditions can be easily avoided from being satisfied, by making slight modifications in underlying spanning tree protocol, to make the underlying network completely secure from the highly treacherous count-to-infinity problem.

When count-to-infinity occurs, the stale information begins to circulate in cycle and thus increments the root path cost of suffering switches with a definite offset, equal to the cycle's path cost, in each complete cycle. Theoretically speaking, count-to-infinity in the network may be temporary or absolute. Temporary count-to-infinity in the network terminates after a definite interval of time. On the other hand absolute count-to-infinity persists forever. Temporary count-to-infinity may occur in a temporarily segregated network, a segregated network that has at least one single rooted alternate port, in which a switch in *orphan subtree* mistakenly turns its orphan alternate port into root or designated port to reunite the segregated network. When this happen count-to-infinity lasts until root path cost of one of the suffering switch exceed to that of the best single rooted alternate port in the network. Absolute count-to-infinity may occur in an absolutely segregated network, a segregated network that has no single rooted alternate port in the network, in which a switch in *orphan subtree* wrongly moves its orphan alternate port in the root port or designated port to reunite the segregated network. As the best single rooted alternate port in the absolutely segregated network has the root path cost of infinity, so count-to-infinity will theoretically last forever.

Both absolute and temporary count-to-infinities are highly undesirable because they adversely effects the convergence time, and thus decreases network availability. They also lead to network congestion and packet loss. Count-to-infinity may induce forwarding loops [15] that results in further increase in the network congestion.

Backup port can be made designated port after failure of its corresponding designated port without any count-to-infinity into the network. The reason is two folded. First, all the root ports on the shared medium start to pretend like single rooted alternate ports that can provide a path to Root Switch through the backup port corresponding to the failed designated port. Second, the root path cost of these pretending single rooted alternate ports is better than that of all orphan alternate ports in the *orphan subtree* i.e. violation of condition 1 of six conditions required for count-to-infinity. Change in port cost of the root port of a switch also forces the port to act like a single rooted alternate port.

Elmeleegy et al. claimed in [5, 15] that injection of stale cached information at alternate port, because of declaring an orphan alternate port as the root port, may cause count-to-infinity. The above discussion further extends this claim by mentioning that injection of stale cached information of the root port of a switch into the network, through a designated port of the switch forming due to retirement of an orphan alternate port, also have potential to induce count-to-infinity into the network (see section 4 for illustrative elaboration).

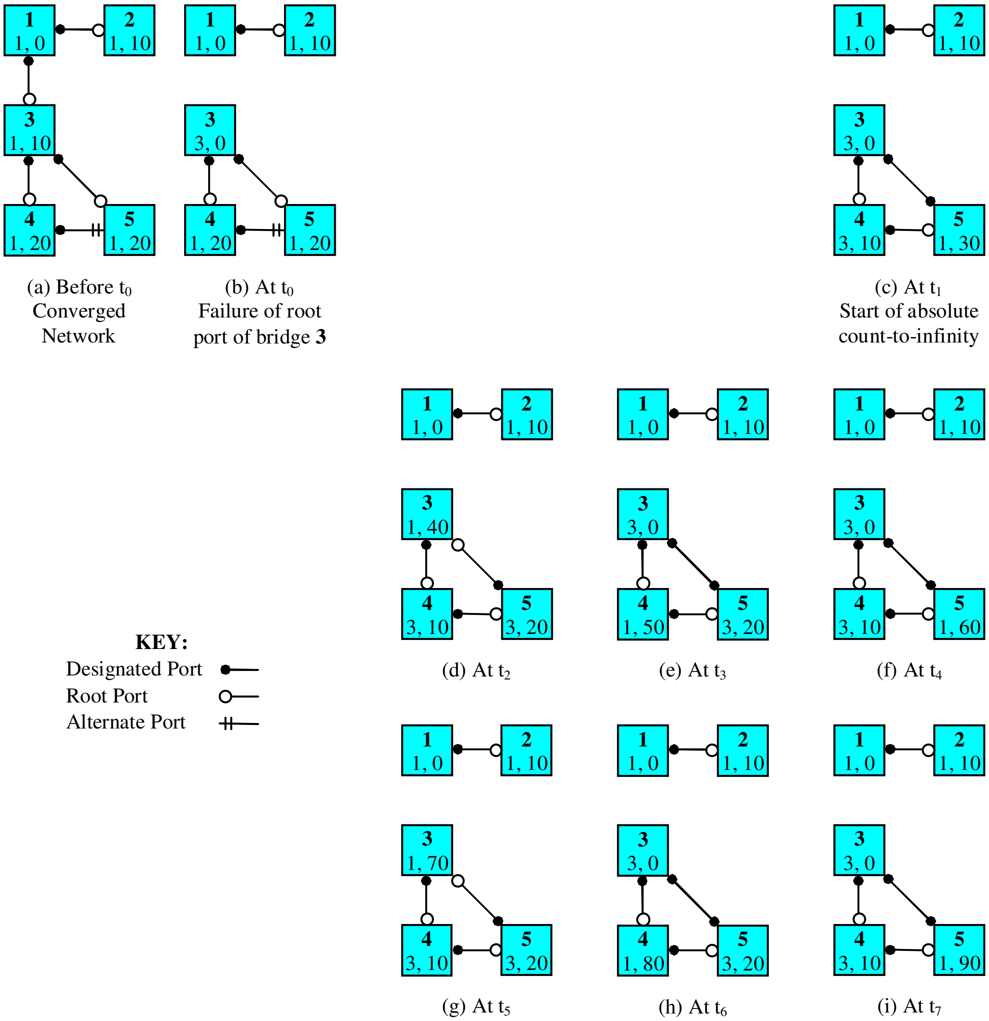
4. COUNT-TO-INFINITY IN RSTP CONTROLLED ETHERNET NETWORKS

RSTP [1] is specifically designed to minimize the convergence time of Ethernet networks. To achieve this goal, RSTP switches uses cached information after an event of failure. But they perform no check to determine whether the received or cached information is fresh (valid) or stale (invalid). This aggressive and optimistic behavior of RSTP switches makes the underlying network highly vulnerable to count-to-infinity problem. This section will explain count-to-infinity in RSTP controlled network.

To illustrate the problem, I will give four specific examples and relate their behaviors to clauses in the IEEE 802.1D (2004) [1] standard. The 7 relevant rules that govern the operation of RSTP that are identified from the IEEE 802.1D (2004) [1] standard are given below.

1. A switch declares itself Root Switch if it perceives itself as the best switch of the network. This will happen if the switch has recently joined a network or it losses its current root port and it has no alternate port. (Clause 17.6).
2. Switches send its own Bridge Protocol Data Unit (BPDU) at regular intervals to guard against loss and to assist in the detection of failed components (LAN, switches and switch ports). (Clause 17.8).
3. A switch immediately transmits its own BPDU on its designated ports if the information it conveys has been changed i.e. when it believes the root has changed or its cost to the root has changed. (Clause 17.8).
4. A switch ages out a received BPDU after three consecutive misses. This is only if the switch cannot physically detect its failure. (Clause 17.21.23).
5. Switch assigns a port role to its each and every port as follows (Clause 17.7):
    a. A port becomes root if it is receiving the best BPDU.
    b. A port becomes alternate if it receives a superior BPDU from another switch and it is not root.
    c. A port becomes designated if receiving BPDU is inferior.
    d. A port becomes backup if it receives a superior BPDU from another port of this switch.
6. An alternate port of a switch can be immediately moved into forwarding state if its current root port has lost its status. (Clauses 17.10).

7. An arrived BPDU can be accepted if and only if it is better (numerically less) or it is from same designated switch and same designated port as that of receiving port's port priority vector. (Clause 17.6).



(a) Before t_0 Converged Network

(b) At t_0 Failure of root port of bridge **3**

(c) At t_1 Start of absolute count-to-infinity

(d) At t_2

(e) At t_3

(f) At t_4

(g) At t_5

(h) At t_6

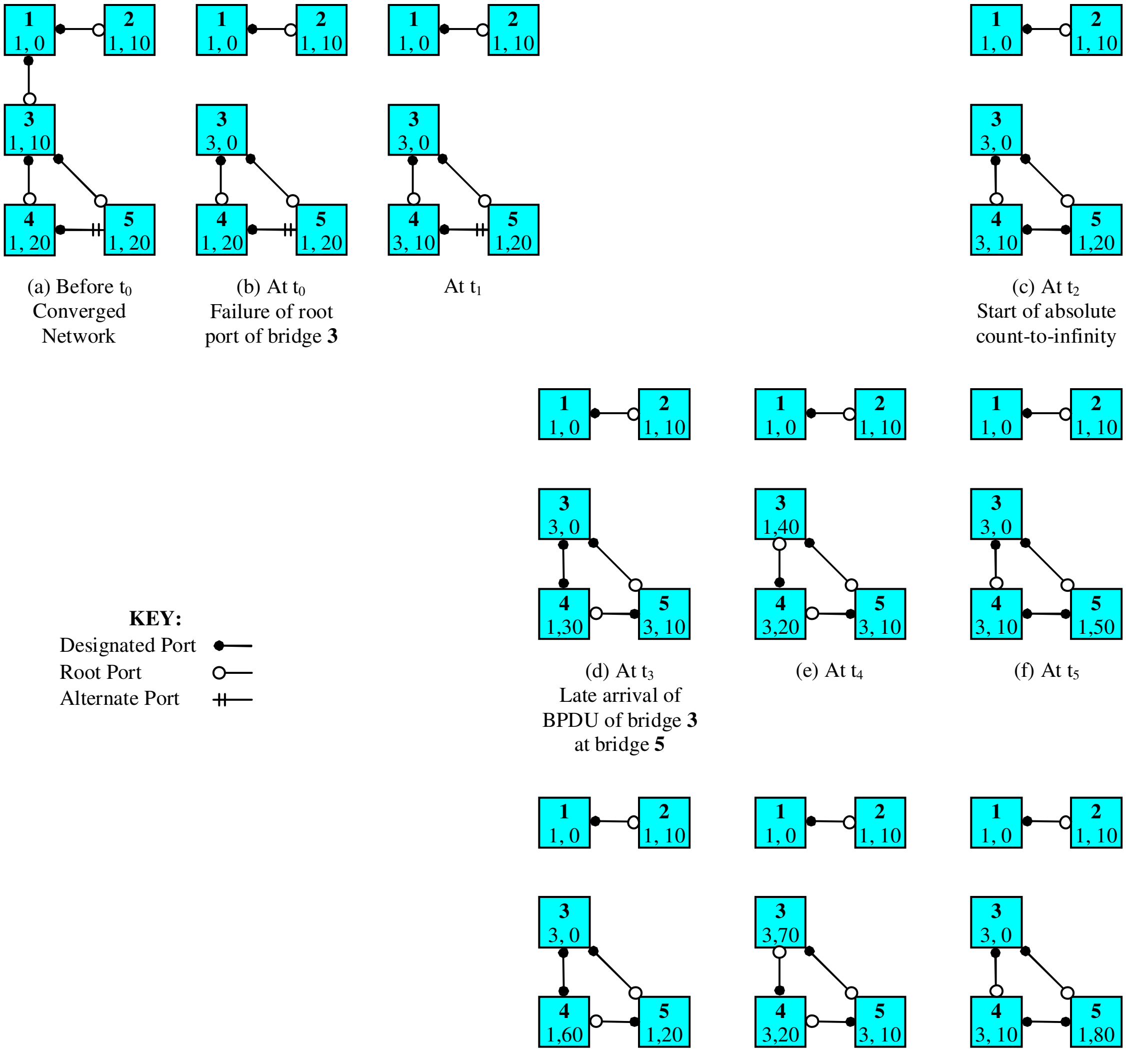
(i) At t_7

FIGURE 2: A network suffering from absolute count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as the new root port.

Now consider the network of switches shown in Figure 2. All links have cost of 10. Figure 2(a) shows the converged network before time t_0. At time t_0 the root port of switch 3 has failed (see figure 2(b)). This port failure divides the network into rooted and orphan subtrees. So switch **1** and **2** are in *rooted subtree* whereas switch **3**, **4** and **5** are in *orphan subtree*.

At time t_0, switch **3** performs the following actions (see figure 2(b));

1. As it realizes its root port has failed, it elects itself as the Root Switch since it has no alternate port (rule 1).
2. Immediately sends an inferior BPDU with itself as the Root Switch on all its designated ports (rule 3).



(a) Before t_0 Converged Network

(b) At t_0 Failure of root port of bridge **3**

At t_1

(c) At t_2 Start of absolute count-to-infinity

(d) At t_3 Late arrival of BPDU of bridge **3** at bridge **5**

(e) At t_4

(f) At t_5

(g) At t_6

(h) At t_7

(i) At t_8

FIGURE 3: A network suffering from absolute count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as designated port.

Note that now both switch **4** and switch **5** are in an absolute inconsistent state because they still believe switch **1** as the Root Switch.

At time t_1, switch **4** takes the following actions (see figure 2(c));

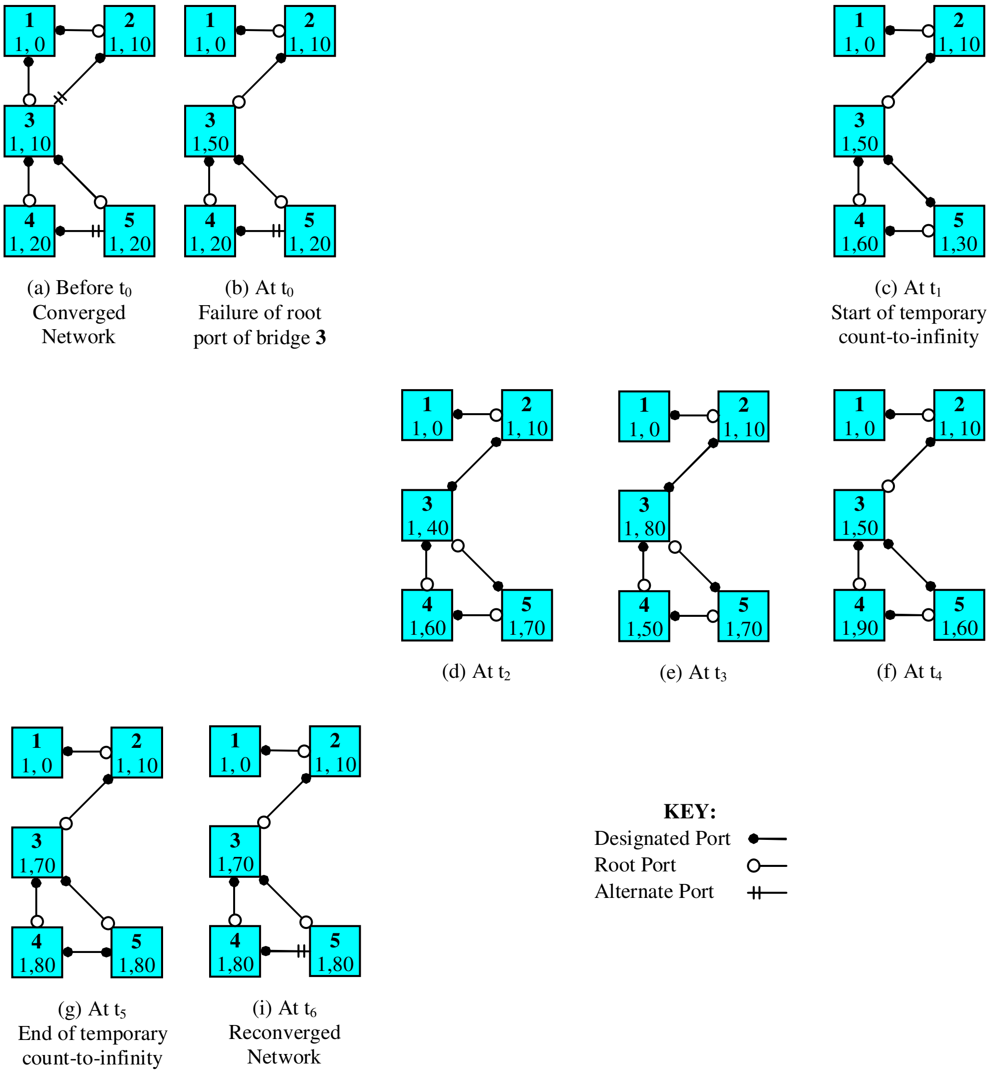
1. Receives and accepts the inferior BPDU from switch 3 (rule 7).
2. Reelects its port to switch 3 as its root port but this time with switch 3 as the root (rule 5 a).
3. Immediately sends an inferior BPDU with switch 3 as root on its designated port (rule 3).

At time t_1, switch **5** executes the following tasks (see figure 2(c));

1. Receives and accepts the inferior BPDU from switch **3** (rule 7).
2. Incorrectly turns its orphan alternate port (a port connected to switch **4**) into root port to the now inaccessible root i.e. switch **1** (rule 6). This is because since at time t_1 switch **5** is in the effective inconsistent state.
3. Switch **5** injects the invalid information of its orphan alternate port into the network (rule 3) by sending the BPDU on its now designated port (a port connected to switch **3**) and thus initiate the count-to-infinity.

At time t_2, switch **5** performs the following actions (see figure 2(d));

1. Receives and accepts the inferior BPDU from switch **4** results in the end of effective inconsistent state (rule 7).
2. Reelects its port to switch **4** as its root port but this time with switch **3** as the root (rule 5 a).
3. Immediately sends this fresh but inferior BPDU on its designated port (rule 3).



(a) Before t_0 Converged Network

(b) At t_0 Failure of root port of bridge **3**

(c) At t_1 Start of temporary count-to-infinity

(d) At t_2

(e) At t_3

(f) At t_4

(g) At t_5 End of temporary count-to-infinity

(i) At t_6 Reconverged Network

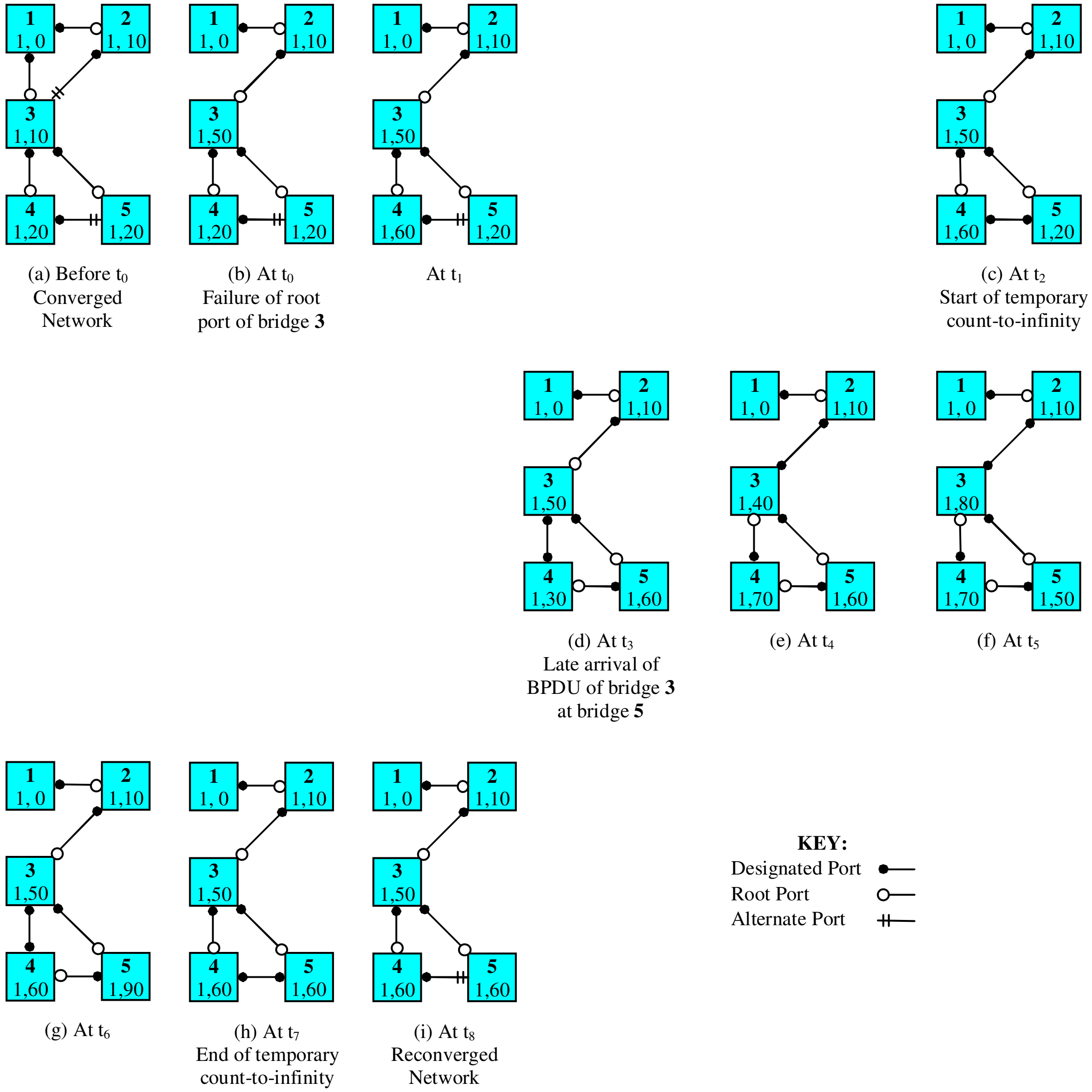
FIGURE 4: A network suffering from temporary count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as the new root port. Link between switch 2 and switch 3 has cost 40 whereas all other links have cost 10.

At time t_2, switch **3** performs the following actions (see figure 2(d));

1. Receives and accepts the stale BPDU from switch **5** (rule 7).
2. Reelects its port to switch **5** as its root port with switch **1** as the root (rule 5 a).
3. Immediately sends the stale BPDU on its designated port (rule 3).

For the rest of time stale BPDU with switch **1** as root, and fresh BPDU with switch **3** as root will chase each other.

Count-to-infinity may also occur in the considered network if switch **5** turns its orphan alternate port into designated port using rule 5 c. It is illustrated in figure 3. This will happen when the switch **5** receives switch **3**'s root port failure information on its alternate port (port connected to switch **4**) before it receives this information on its root port (port connected to switch **3**).



(a) Before t_0
Converged
Network

(b) At t_0
Failure of root
port of bridge **3**

At t_1

(c) At t_2
Start of temporary
count-to-infinity

(d) At t_3
Late arrival of
BPDU of bridge **3**
at bridge **5**

(e) At t_4

(f) At t_5

(g) At t_6

(h) At t_7
End of temporary
count-to-infinity

(i) At t_8
Reconverged
Network

FIGURE 5: A network suffering from temporary count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as designated port. Link between switch 2 and switch 3 has cost 40 whereas all other links have cost 10.

Rule 3, rule 5 c, rule 6 and rule 7 of RSTP play a vital role in inducing absolute count-to-infinity into the network. Rule 3 allows a switch to rapidly propagate the information of root port failure to downstream switches through its designated ports. On the other hand, rule 7 forces the downstream switches to accept this failure information. Moreover, when a switch receives this failure information it may turns its alternate port into root or designated port, even when it is in inconsistent state, on the bases of its invalid cached information (rule 5 c and rule 6). As a result count-to-infinity may induce into the network.

RSTP is also susceptible to temporary count-to-infinity. Figure 4 is showing a network that suffers from temporary count-to-infinity because switch 5 is declaring its orphan alternate port as new root port. Where as figure 5 is showing the same network suffering from temporary count-to-infinity as switch 5 is announcing its orphan alternate port as designated port. Rule 3, rule 5 c, rule 6 and rule 7 that play vital role in induction of absolute count-to-infinity are also responsible for temporary count-to-infinity. Same lines of reasoning that are used for explaining absolute count-to-infinity in RSTP can also be used for temporary count-to-infinity.

In summary, RSTP [1] is vulnerable to both absolute and temporary count-to-infinities. The reason is two folded. First, RSTP switches have tendency to use their alternate ports, to rapidly converge the network, even when they are in effective inconsistent state and so may inject stale (invalid) information into the network through their orphan alternate ports or through their retiring root ports. Second, RSTP switches cannot distinguish between stale (invalid) and fresh (valid) information (BPDU) and so stale information may last unnoticeably into the network for long time. This undesirable behavior of RSTP leads to unpredictable convergence time that may as high as tens of seconds [5], [14] and [15]. Count-to-infinity may also induce forwarding loop in RSTP controlled network that lead to network-wide congestion and packet loss as explained in [15].

5. DRSTP: THE DELAY RAPID SPANNING TREE PROTOCOL

Delay Rapid Spanning Tree Protocol – DRSTP – is an extension to RSTP. It is designed specifically to ensure that an Ethernet network converge as quickly as possible, after a link, port or switch failure, without inducing count-to-infinity into the network. The best thing about this solution is that it is completely backward compatible to legacy RSTP\STP switches.

DRSTP prevents count-to-infinity problem in mixed environment by simply forcing DRSTP switches to postpone transmission of BPDUs on recently retiring root or alternate port during the estimated period of effective inconsistence. Moreover, DRSTP switches also defer to transmit better BPDUs received from legacy switches for time equal to estimated period of effective inconsistence. This is to ensure that stale better BPDUs transmitted by legacy switches will not spoil the network. The next subsection will drive mathematically the estimated period of effective inconsistence. It is noteworthy that period of effective inconsistence for a bridge usually last for only few hundreds of microseconds in most cases.

Derivation for Draining Out Stale BPDUs

In RSTP [1] cost of a link, by default, is inversely proportional to the bandwidth of the link and thus represents the time to transmit single bit on the link. Mathematically,

$$t = kc \qquad (1)$$

where, t is transmission time of single bit,
c is the cost of the link and
k is the constant of proportionality and it is, by default, equal to 0.05 picoseconds according to [1].
So the transmission time $T_{transmission}$ of a “n bits BPDU” is

$$T_{transmission} = nt$$
$$T_{transmission} = nkc \qquad (2)$$

Total time T_{total} taken by BPDU can be defined as

$$T_{total} = T_{transmission} + T_{propagation} \qquad (3)$$

where, $T_{propagation}$ is time taken by BPDU to travel through the link.
But, in Ethernet $T_{propagation}$ is negligible, so

$$T_{total} \approx T_{transmission}$$
$$T_{total} = nkc \qquad (4)$$

Consider a network of switches as shown in figure 6. Let **R** be the Root Switch of that network. Consider a switch **F** such that T_f be the total time taken by a BPDU send by Root Switch **R** to reach to switch **F**. Consider another switch **B** of the network. Let T_r and T_a be the time taken a BPDU, send by Root Switch **R**, take to reach the root port *r* and the alternate port *a* of switch **B** respectively. Suppose c_r and c_a be the root path cost of the root port *r* and alternate port *a* of switch **B** respectively.

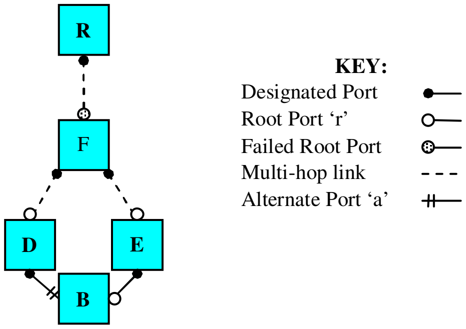


FIGURE 6: Network of switches used for deriving period of effective inconsistence for switch B.

So,

$T_r = nkc_r$ (5)

and

$T_a = nkc_a$ (6)

Suppose the root port of switch **F** fails. So the switch **F** sends a BPDU, announcing switch **F** as Root Switch, on all of its designated ports. It put switch **B** in *orphan subtree* and so in inconsistent state. Let T_1 and T_2 be the total time taken by the BPDU send by switch F to reach to the root port *r* and the alternate port *a* of switch B respectively. Hence, the period of effective inconsistence ΔT for switch **B** can be defined as the difference of time T_2 to time T_1 i.e.

$\Delta T = T_2 - T_1$ (7)

But

$T_r = T_f + T_1$ (8)

and

$T_a = T_f + T_2$ (9)

Using (5),(6),(8) and (9)

$\Delta T = nk(c_a - c_r)$ (10)

Since, *a* is the alternate port of switch **B**, so the following relation will hold:

$c_a - c_p \leq c_r \leq c_a$ (11)

where c_p is port path cost of alternate port *a* of switch **B**

Therefore, using (11)

$\Delta T \leq nkc_p$ (12)

So, the estimated value of inconsistent port timer D, see section 5.2 for definition inconsistent port timer D, must be:

$D = nkc_p + t_{processing} + C$ (13)
where, $t_{processing}$ is the average BPDU processing time
and C is the additive constant to handle variations in ΔT.

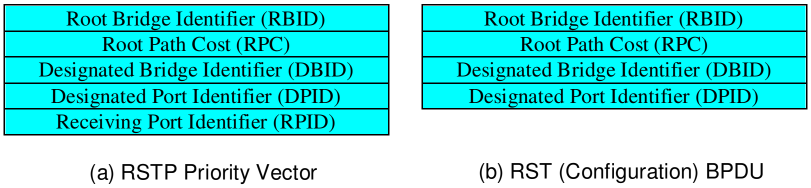
Moreover, if stale BPDU is injected by a switch in effective inconsistent state, it can be at most nkc_p unit of time ahead of fresh BPDU. Hence, the estimated value of inconsistent port timer *D* can also be use as that of count-to-infinity suppression timer *S*, see section 5.2 for definition count-to-infinity suppression timer *S*.

For network having slowest link of 10Mbps, average BPDU processing time $t_{processing}$ of 48.8µs and additive constant *C* of 50µs, the estimated values for inconsistent port timer *D* and count-to-infinity suppression timer *S* are no more than 150µs. These are very small and quite acceptable values.

However, the above derivation is valid only with assumption that all links in the network have default cost and no BPDU loss is occurring. More careful network analysis is need for networks using non-default link cost to make good estimation of values of inconsistent port timer *D* and count-to-infinity suppression timer *S*. However, it is expected that the two timers' value remains low for most commercial networks even when they are not using default link cost.

Protocol Definition

Like RSTP [1], operation of DRSTP can be defined precisely with the help of priority vectors. Figure 7 is showing the structure of an RSTP Priority Vector and RST (Configuration) BPDU respectively.



Root Bridge Identifier (RBID)
Root Path Cost (RPC)
Designated Bridge Identifier (DBID)
Designated Port Identifier (DPID)
Receiving Port Identifier (RPID)

(a) RSTP Priority Vector

Root Bridge Identifier (RBID)
Root Path Cost (RPC)
Designated Bridge Identifier (DBID)
Designated Port Identifier (DPID)

(b) RST (Configuration) BPDU

FIGURE 7: Structure of RSTP Priority Vector and RST BPDU.

In detail, the DRSTP modifies the RSTP as follows:

1. An DRSTP switch associates two timers with each switch's port namely inconsistent port timer and count-to-infinity suppression timer.
2. In an DRSTP switch, a port is not eligible to transmit BPDUs, when inconsistent port timer is running on that port.
3. In an DRSTP switch, a port cannot participate in the root port election, if count-to-infinity suppression timer is running on it. So, such ports cannot become the root port.
4. An DRSTP switch divides received RST (Configuration) BPDUs into four distinct types namely Better RST (Configuration) BPDU, Repeated RST (Configuration) BPDU, Inconsistent RST (Configuration) BPDU and Worse RST (Configuration) BPDU. It is in contrast to RSTP which divides receiving BPDU into only three major types i.e. superior BPDU, repeated BPDU and inferior BPDU.
5. An DRSTP switch considers a received RST (Configuration) BPDU as Better RST (Configuration) BPDU if it is better (numerically less) than currently stored BPDU (Port Priority Vector).
6. An DRSTP switch handles a received RST (Configuration) BPDU as Repeated RST (Configuration) BPDU if it is same (numerically equal) as currently stored BPDU (Port Priority Vector).

7. An DRSTP switch treats a received RST (Configuration) BPDU as Worse RST (Configuration) BPDU if it is worse (numerically greater) than currently stored BPDU (Port Priority Vector) but it is not received from previous source. A receiving BPDU is said to be received from previous source if its Designated Bridge Identifier (DBID) and Designated Port Identifier (DPID) are equal to that of Port Priority vector of receiving port.
8. An DRSTP switch believes that the received RST (Configuration) BPDU is an Inconsistent RST (Configuration) BPDU if it is worse (numerically greater) than currently stored BPDU (Port Priority Vector) and it is received from previous source.
9. In DRSTP switch, a port starts its count-to-infinity suppression timer when it receives a better RST (Configuration) BPDU from a legacy RSTP (STP).
10. In DRSTP switch, a port starts its inconsistent port timer if it receives an Inconsistent RST (Configuration) BPDU.
11. DRSTP should be assigned a new protocol version. It enables DRSTP switches to differentiate between RST BPDUs transmitted by legacy RSTP switches and RST BPDUs transmitted by DRSTP switches.

Discussion

An STP switch discards an Inconsistent BPDU, a worse BPDU from previous source. This is the major cause of slow convergence of STP. Whereas, an RSTP switch handles an Inconsistent BPDU as if it were a better BPDU. But this behavior of RSTP switches makes it vulnerable to count-to-infinity problem. In contrast, an DRSTP switch considers an Inconsistent BPDU as a marker for beginning of effective inconsistent state. So, a port of an DRSTP switch starts its inconsistent port timer when it receives an Inconsistent BPDU. It prevents the port from injecting probably stale BPDUs through retiring alternate or root port and thus making violation of condition 3 of six conditions required for count-to-infinity. A port in an DRSTP switch starts its count-to-infinity suppression timer if it receives a Better BPDU from a legacy switch. A port running count-to-infinity suppression timer is not allowed to participate in root port election. This is because legacy switches have a tendency of injecting stale BPDUs when they are in effective inconsistent state. By disallowing a port to participate in root port election, the switch ensures violation of condition 4 of six conditions required for count-to-infinity.

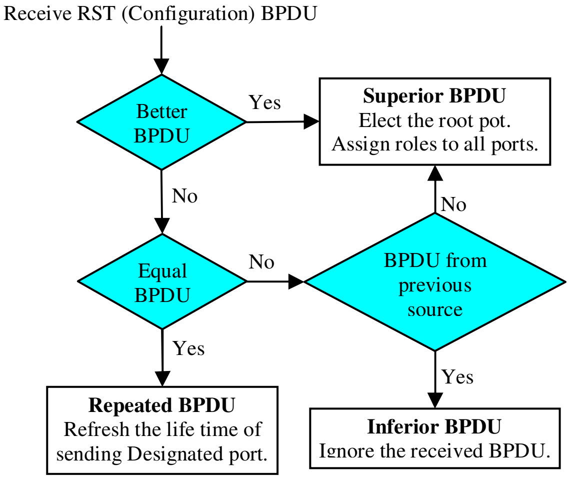

Receive RST (Configuration) BPDU
Better BPDU
Yes
Superior BPDU
Elect the root pot.
Assign roles to all ports.
No
No
Equal BPDU
No
BPDU from previous source
Yes
Yes
Repeated BPDU
Refresh the life time of sending Designated port.
Inferior BPDU
Ignore the received BPDU.

FIGURE 8: Processing of received RST (Configuration) BPDU in RSTP.

DRSTP is backward compatible to both STP and RSTP. But DRSTP cannot prevent count-to-infinity in the presence of legacy STP switches in the network. This is because STP switches discards Inconsistent BPDUs and so making the period of effective inconsistence considerably high i.e. in the order of tens of seconds. Moreover, DRSTP is a count-to-infinity prevention technique, so count-to-infinity may occur in the network even in the presence of DRSTP switches.

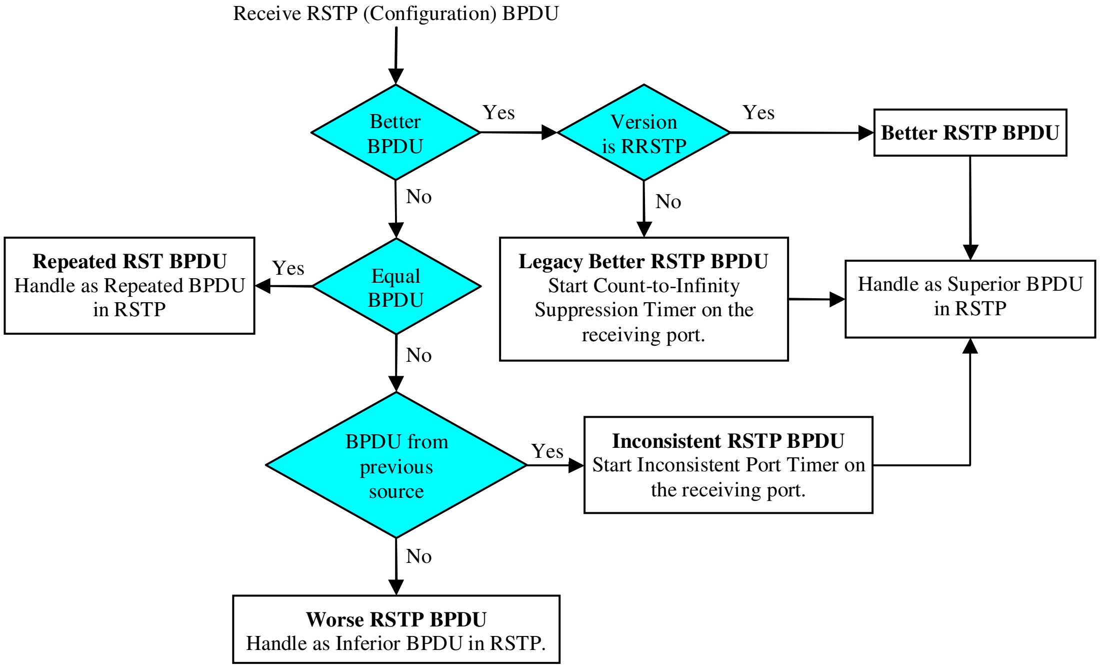

Receive RSTP (Configuration) BPDU
Better BPDU
Yes
Version is RRSTP
Yes
Better RSTP BPDU
No
No
Repeated RST BPDU
Handle as Repeated BPDU in RSTP
Yes
Equal BPDU
Legacy Better RSTP BPDU
Start Count-to-Infinity Suppression Timer on the receiving port.
Handle as Superior BPDU in RSTP
No
BPDU from previous source
Yes
Inconsistent RSTP BPDU
Start Inconsistent Port Timer on the receiving port.
No
Worse RSTP BPDU
Handle as Inferior BPDU in RSTP.

FIGURE 9: Procession of received RST (Configuration) BPDU in DRSTP.

Figure 8 is showing the processing of received RST (Configuration) BPDU by a legacy RSTP switch. Whereas, figure 9 is showing the processing of received RST (Configuration) BPDU by an DRSTP switch. Handling of port failure in RSTP and DRSTP is shown in figure 10.

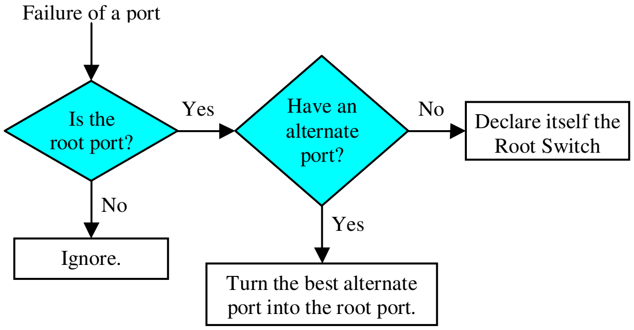

Failure of a port
Is the root port?
Yes
Have an alternate port?
No
Declare itself the Root Switch
No
Ignore.
Yes
Turn the best alternate port into the root port.

FIGURE 10: Handling of port failure in RSTP and DRSTP.

Comparison With Contemporary Protocols

This section will critically discuss DRSTP with other contemporary protocols. The four other protocols that will be used for comparison are STP [7], RSTP [1], RSTP with Epoch [5][15] and Ether Fuse [6]. The five key aspects that will be discussed during comparison are vulnerability against count-to-infinity, convergence time, protocol implementation, extra hardware requirement, and backward compatibility.

Both STP [7] and RSTP [1] are susceptible to temporary and absolute count-to-infinities. In contrast, DRSTP provide protection, to some extend, against both type of count-to-infinities.

“RSTP with Epoch” is a new protocol that specifically designed to address the count-to-infinity problem but unfortunately it is vulnerable against temporary count-to-infinity.

STP exhibits very slow convergence time of up to 50s [2]. In contrast, RSTP may converge with in 1-3s due to its aggressive and optimistic approach. But this low convergence time is showed by RSTP only in absence of count-to-infinity. In contrast, DRSTP is expected to usually exhibit convergence time of 1-3s. Convergence time of RRSTP with Epoch is also comparable to that RSTP.

DRSTP is completely backward compatible to RSTP. It is also compatible to STP but at the expense of exposure to count-to-infinity. “RSTP with Epoch” is also backward compatible to legacy switches. But it does not ensure protection against count-to-infinity in mixed environment having legacy switches.

Ether Fuse [6] is a solution proposed by Elmeleegy et al. to protect network of legacy switches from adverse effects of count-to-infinity. Ether Fuse [6] uses a reactive approach to the problem that is at first it allows count-to-infinity to occur but stops it before it become severe. This approach of Ether fuse toward the problem is in sharp contrast with other protocols as they use a proactive approach. Ether Fuse is a completely standalone solution that has its own memory and hardware requirement. In contrast, DRSTP neither require extra memory nor hardware for deployment. Further, DRSTP can be built very easy and quickly using already available RSTP modules. In fact, DRSTP require subtle changes in only three state machine of RSTP namely Role Selection State Machine, Port Information State Machine and Transmit State Machine.

Hence DRSTP can be considered as an easy to implement backward compatible solution to reduce the occurrences of count-to-infinity in spanning tree controlled Ethernet networks with very little compromise on convergence time due to insertion of a very small delay of few hundred microseconds. As it is decreasing the frequency of cont-to-infinity, so the overall reliability of Ethernet networks will increase considerably.

		STP	RSTP	DRSTP	Ether Fuse	RSTP with Epoch
Frequency of Count-to-infinity	Temporary	High	High	Low	--	High
	Absolute	High	High	Low	--	Zero
Convergence time	In case of no count-to-infinity	Up to 50s	1-3s	1-3s	--	Order of round trip time to Root Switch
	In case of count-to-infinity	Order of maximum message age	Order of maximum message age.	Order of maximum message age.	--	Order of maximum message age
Approach to handle count-to-infinity		N/A	N/A	proactive	reactive	proactive
Backward compatibility		N/A	Yes	Yes	Yes	Yes

TABLE1: Comparison of DRSRP with other contemporary protocols.

6. RELATED WORK

Reliability and scalability of Ethernet are main concerns for researchers for last two decades. Some researchers believe that reliability of Ethernet can be enhanced by use of link state routing protocols. One of such attempts is Rbridges that is proposed by Perlman [16]. Garcia et al. also proposed use of link state routing to substitute spanning tree [17].

Turn-prohibition is another technique used in Ethernet to improve scalability and reliability. Up/Down proposed by Schroeder et al. [18], Turn Prohibition (TB) proposed by Starobinski et al. [19], Tree-Based Turn-Prohibition (TBTP) proposed by Pellegrini et al. [20] and Hierarchal Up/Down Routing and Bridging Architecture (HURP/HURBA) proposed by Ibáñez et al. [21] are few well-known algorithms based on this technique.

SEATTLE proposed by Kim et al. [22] is a completely new layer 2 network architecture. However, it is not a backward compatible solution. Sharma et al. [23] introduce a multiple spanning tree architecture that improves the throughput and reliability over when using a single spanning tree. SmartBridges [24] uses the techniques of diffusing computation [25] and effective global consistency to achieve loop-freeness.

Instead of using other techniques, “RSTP with Epochs” proposed by Elmeleegy et al. [5] and [15] made an effort to increase reliability of spanning tree itself. It extends RSTP [1] to eliminate count-to-infinity. Unfortunately “RSTP with Epochs” [5] and [15] has no ability to handle count-to-infinity in mixed environment. Moreover, it cannot tackle temporary count-to-infinity problem even in full environment. DRSTP, an extension of RSTP, tries to mitigate count-to-infinity problem in mixed environment having legacy switches. It is completely backward compatible because it proposes changes only in interpretation of received BPDU.

7. CONCLUSION & FUTURE PLAN

This paper presents classical count-to-infinity problem in a novel fashion and point out that count-to-infinity can be temporary or absolute in a spanning tree controlled network. The paper then shows that RSTP [1] is susceptible to both temporary and absolute count-to-infinity. Spanning tree protocols like RSTP [1] that are exposed to count-to-infinity problem exhibit poor convergence, depending upon how long count-to-infinity situation persist. This paper also proposes a simple and effective solution – named as Delay Rapid Spanning Tree Protocol – to mitigate to count-to-infinity problem in RSTP. To achieve his goal, DRSTP inserts a small delay of few hundred microseconds before injecting its own cached information on recently retiring alternate or root port. Moreover, DRSTP hesitates to use a port as the root port for a small period of time when it is receiving Better BPDUs from legacy switches. Hence, it is expected that the solution will significantly enhance the dependability of Ethernet network without compromising much on its availability.

My future plan is to design a spanning tree protocol that will provide guaranteed protection against both absolute and temporary count-to-infinities.

8. ACKNOWLEDGEMENT

I would like to express gratitude to my parents for their unconditional support. I would also like to acknowledge the efforts of cooperative team of IJCN in making my maiden publication possible.

9. REFERENCES

1. LAN/MAN Standards Committee of the IEEE Computer Society. *“IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges - 802.1D”*. 2004.

2. Cisco Systems, Inc. *“Spanning Tree Protocol Problems and Related Design Considerations”*. Available: www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a00800951ac.shtml

3. Cisco Systems, Inc. *“Spanning-Tree Protocol Enhancements using Loop Guard and BPDU Skew Detection Features”*. Available: www.cisco.com/warp/public/473/84.html

4. Cisco Systems, Inc. *“Understanding and Configuring the Unidirectional Link Detection Protocol Feature”*. Available: www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a008009477b.shtml

5. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *“On Count-to-Infinity Induced Forwarding Loops in Ethernet Networks”*. In IEEE Infocom 2006.

6. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *“EtherFuse: An Ethernet Watchdog”*. In ACM SIGCOMM 2007.

7. R. Perlman. “*An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN”*. In the proceedings of 9th ACM Data Communications Symposium. New York, USA, 1985.

8. M Seaman. *“High Availability Spanning Tree”*. Available: www.ieee802.org/1/files/public/docs1998/hasten7.pdf.

9. M. Seaman. *“Speedy Tree Protocol”*. Available: www.ieee802.org/1/files/public/docs1999/speedy_tree_protocol_10.pdf.

10. M. Seaman. *“Truncating Tree Timers”*. Available: www.ieee802.org/1/files/public/docs1999/truncating_tree_timing_10.pdf.

11. V. Jain and M. Seaman. *“Faster flushing with fewer addresses”*. Available: www.ieee802.org/1/files/public/docs1999/faster_flush_10.pdf.

12. G. Malkin. *“RIP version 2”*. RFC 2453. Nov 1998.

13. Cisco Systems, Inc. *“Enhanced Interior Gateway Routing”* Available www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml.

14. Myers, T. E. Ng, and H. Zhang. *“Rethinking the Service Model: Scaling Ethernet to a Million Nodes”*. In 3rd Workshop on Hot Topics in networks. 2004.

15. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *“Understanding and Mitigating the Effects of Count to Infinity in Ethernet Networks”*. IEEE/ACM Transactions on Networking, February 2009.

16. R. Perlman. *“Rbridges: Transparent routing”*. In IEEE Infocom 2004.

17. R. Garcia, J. Duato and F. Silla. *“LSOM: A link state protocol over MAC addresses for metropolitan backbones using optical Ethernet switches”*. In 2nd IEEE International Symposium on Network Computing and Applications. 2003.

18. M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, C. Thacker. *“Autonet: A High-Speed, Self–Configuring Local Area Network Using Point–to–Point Links”*. IEEE Journal on Selected Areas in Communications, 9(8):1318–1335, 1991.

19. D. Starobinski, G. Karpovsky, F. Zakrevsky. *“Applications of network calculus to general topologies”*, IEEE/ACM Transactions on Networking, 11(3):411–422, 2003.

20. F. D. Pellegrini, D. Starobinski, M. G. Karpovsky and L. B. Levitin. *“Scalable cycle-breaking algorithms for gigabit Ethernet backbones”*. In IEEE Infocom 2004.

21. Guillermo Ibáñez, Alberto García-Martínez, Juan A. Carral, Pedro A. González, Arturo Azcorra, José M. Arco. *“HURP/HURBA: Zero-configuration hierarchical Up/Down routing and bridging architecture for Ethernet backbones and campus networks”*, Computer Networks, 54(1):41-56,2010.

22. C. Kim, M. Caesar, and J. Rexford. "*Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises"*. In ACM SIGCOMM. 2008.

23. S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: "A multispanning tree Ethernet architecture for metropolitan area and cluster networks". In IEEE Infocom. 2004

24. T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. *"SmartBridge: A scalable bridge architecture"*. In ACM SIGCOMM. 2000.

25. E. W. Dijkstra, C. S. Scholten. *"Termination detection for diffusing computations"*. Information Processing Letters, 11(1):14, 1980.

MFMP Vs ECMP Under RED Queue Management

Ahmed Redha Mahlous waellena@yahoo.com
Al Imam Mohamed Bin Saud University
Riyadh, Saudi Arabia

Brahim Chaourar bchaourar@hotmail.com
Al Imam Mohamed Bin Saud University
Riyadh, Saudi Arabia

Abstract

In this paper we compare the performance of a Maximum Flow Multi Path routing (MFMP) and Equal Cost Multi Path routing (ECMP) under a congestion avoidance scheme: Random Early Detection (RED). We show through simulation that MFMP performs well than ECMP in terms of mean end to end delay, packet loss percentage and packet delivery percentage.

Keywords: Computer Network Routing, Maximum Flow, Shortest Path, Multi Paths, RED.

1. INTRODUCTION

The current Internet protocols such as RIP [11], OSPF [13] and BGP [19] use a single shortest path to forward traffic from any source to any destination in the network.

Even that some protocols such as OSPF and EIGRP support a multipath routing, however it is not used until configured manually. The use of solely of shortest path can lead to unbalanced traffic distribution due to congestion of the frequently used shortest path. To overcome this problem, a multipath routing is proposed as an alternative to single shortest path to take advantage of network redundancy, distribute load [15], improve packet delivery reliability [8], ease congestion on a network [1],[7], improve robustness [16], increase network security [3] and address QoS issues [4].

In literature we find several multipath schemes both in wired and wireless routing [17], [2], [20], [24–30] however the most used multipath routing in today's router is the Equal Cost Multi Path [13], [31], [32].

In our previous work [9] we have proposed a multipath algorithm that is based on a Maximum Flow algorithm and we have shown [10] that our proposed algorithm (MFMP) is better than Equal Cost Multi Path (ECMP) when using First In First Out (FIFO) queuing discipline in terms of packets delivery and delay.

In this paper we use a congestion avoidance scheme, Random Early detection (RED) to look deeper at the performance of MFMP over ECMP.

The remaining of the paper is organized as follows. In Section 2, we give a short description of RED, MFMP and ECMP. In Section 3, we give the parameters of the simulation. In Section 4, we present the results of the simulation and analyze them. And finally we conclude in Section 5.

2. RED, MFMP and ECMP

RED [5] was designed with the objectives to minimize packet loss and queuing delay, avoid global synchronization of sources, maintain high link utilization, and remove biases against bursty sources. The basic idea behind RED queue management is to detect incipient congestion early and to convey congestion notification to the end-hosts, allowing them to reduce their transmission rates before queues in the network overflow and packets are dropped. To do this, RED maintains an exponentially-weighted moving average (EWMA) of the queue length which it uses to detect congestion. When the average queue length exceeds a minimum threshold (minth), packets are randomly dropped or marked with an explicit congestion notification (ECN) bit [6]. When the average queue length exceeds a maximum threshold (maxth), all packets are dropped or marked. MFMP sends packets over multiple paths obtained by a maximum flow algorithm, while ECMP uses multiple shortest paths. Both algorithms in this simulation use RED to control congestion in the network nodes.

3. SIMULATION ENVIRONMENT

Using OMNET++ 3.3, the performance of MFMP is compared to ECMP in the topology shown in Figure 1, which represents the optical core of the infrastructure in the COST-239 project [14]. This core network can be represented as a graph G = (N, L), where N represents a set of nodes interconnected by a set of links L as shown in Figure 2. It consists of 11 nodes (routers), of which, one bursty source (node 1) and one sink (destination) (node 9) and 25 bidirectional links of different weight as shown in Figure 2.

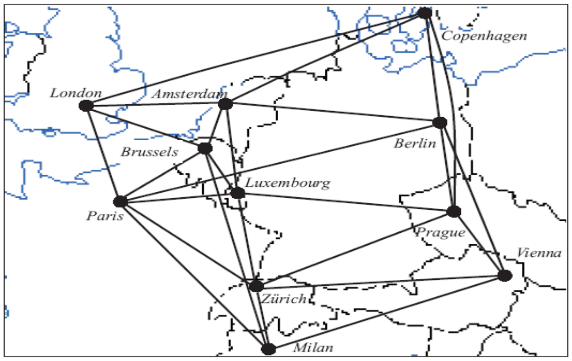


FIGURE 1: The COST-239 core network

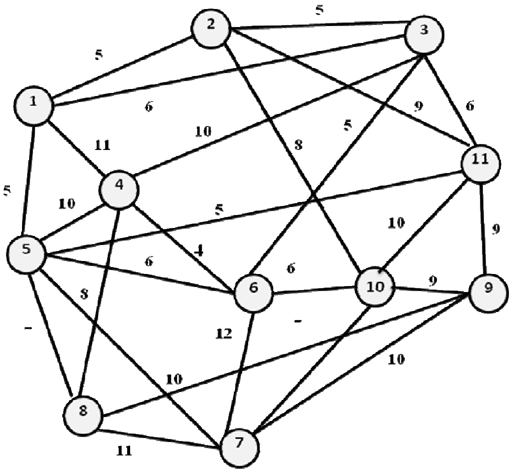


FIGURE 2: Graph representation of COST-239 core network

The TCP packets used in our simulation are of size equal to 1500 bytes. We used the same packets size for MFMP and ECMP to ensure that, packets are treated fairly by the routers for each protocol with regards to the size of the packets.

We used only unidirectional traffic. That is the source sends the data packets to the sink and the receiver sends nothing except ACK packets back on the reverse path. This approach has been followed by many researchers in their simulation work in order to avoid what is known ACK compression [12], [22] and [23].

The traffic is generated by one ON-OFF source that sends bursts with random duration distributed by Pareto distribution to model self-similar arrival and to model a broadband traffic [18]. To consider the effect of variation of load, we run our simulation under different traffic intensity scenarios varied from heavy traffic load corresponding to short inter arrival time (2000 packets/s for λ= 0.0005 and 1000 pkts/s for λ= 0.0010) to light traffic load corresponding to long inter arrival time (667 pkts/s for λ= 0.0015, 500 pkts/s for λ= 0.0020, and 400 pkts/s for λ= 0.0025). The ON period follows Pareto distribution (1, 0.5) and OFF period follows an exponential distribution (0.5 s).

The metrics of interest used to evaluate the performance of MFMP Vs ECMP are: packet delivery ratio percentage, mean end to end delay, and packet loss ratio percentage.

We run our simulation under two RED scenarios one for a short queue size (Minth=15, Maxth=33) and another for a medium queue size (Minth=20, Maxth=66).

4. SIMULATION RESULTS AND ANALYSIS

The following graphs summarize the results of the simulation.

Figure 3 - Packets delivery percentage - short queue size scenario

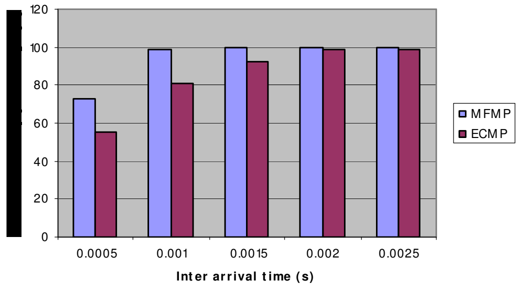


Figure 4 - Packets loss percenatge - short queue size scenario

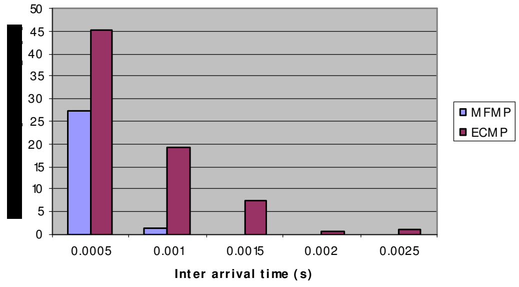


Figure 5 - Packets delivery percentage - meduim queue size scenario

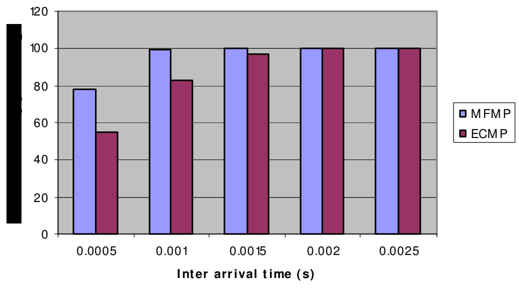


Figure 6 - Packets loss percentage - meduim queue size scenario

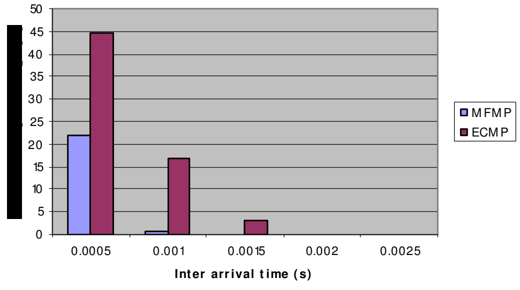


Figure 7 - Mean end to end delay - short queue size scenario

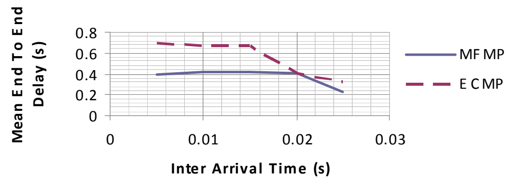
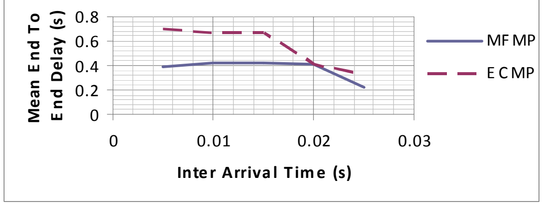


Figure 8 - Mean end to end delay - meduim queue size scenario



The performance of MFMP is clearly better than ECMP (in terms of packets delivery and packets loss percentages and mean end to end delay) for heavy traffic loads (λ = 0.0005, 0.0010, 0.0015)

both in short and medium queue size scenarios. But for light loads (λ = 0.0020, 0.0025), MFMP and ECMP give approximately the same performance because there is no congestion and RED is without effects. The long queue size scenario has not been considered for the same reason.
This performance is due to that when some paths in a multipath are congested, the max flow uses other alternative paths, that can be with same length as paths in ECMP or longer, that maximize the flow. So MFMP is able to benefit from the number of alternative paths. However the topology of the considered network influences theses results: if we have a network where MFMP uses long paths and ECMP short paths, then the performance of our algorithm can be worst than ECMP, at least for some cases of heavy traffic loads. Fortunately, this is not the case in practice. In most of the real networks, the paths of MFMP are the paths of ECMP plus some extra paths. So in the worst case, MFMP and ECMP have the same performance.

5. CONCLUSION

In this paper, we have shown through simulations that MFMP is better than ECMP under RED queue management in terms of packets delivery and loss percentage, and mean end to end delay. This is to confirm our previous results obtained in the case of FIFO queue management scheme.
Future investigations can be done in other queue management schemes.

Acknowledgement
This research is a part of the project SP–14–43 which is supported financially by King Abdulaziz City for Sciences and Technology, Kingdom of Saudi Arabia.

REFERENCES

1. S. Bahk and M. E. Zarki, Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks, Proceedings ACM SIGCOMM 22 (4): 53-64, 1992.

2. R. Banner and A. Orda, Multipath routing algorithms for congestion minimization, IEEE/ACM Transactions on networking 15 (2): 413-424, 2007.

3. S. Bohacek, J. Hespanha, J. Lee, K. Obraczka and C. Lim, Enhancing security via stochastic routing, Proceedings 11th International Conference on Computer Communications and Networks: 58-62, 2002.

4. S. Bohacek, J. Hespanha, J. Lee, C. Lim and K. Obraczka, Game theoretic stochastic routing, IEEE Transactions on Parallel and Distributed Systems 18(9): 1227-1240, 2007.

5. S. Floyd and V. Jacobson, Random Early Detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1(4): 397-413, 1993.

6. S. Floyd, TCP and Explicit Congestion Notification, SIGCOMM Computer Communication Review 24(5): 10-23, 1994.

7. P. Georgatsos and D. Griffin, A management system for load balancing through adaptive routing in multiservice ATM networks, Proceedings IEEE Infocom: 863-870, 1996.

8. K. Ishida, Y Kakuda and T. Kikuno, A routing protocol for finding two node-disjoint paths in computer networks, Proceedings. IEEE International Conference on Network Protocols (ICNP): 340-347, 1992.

9. A. R. Mahlous, R. J. Fretwell and B. Chaourar, MFMP: Max Flow Multipath routing algorithm, Proceedings 2nd UKSIM European Symposium on Advanced Information Networking and Applications Workshops : 482-487, 2008.

10. A. R. Mahlous, B. Chaourar and M. Mansour, Performance evaluation of Max Flow Multipath Protocol with congestion awareness, WAINA: Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops: 820-825, 2009.

11. G. Malkin, RIP version 2 protocol analysis, IETF Internet RFC 1721, November 1994.

12. J. Mogul, Observing TCP dynamics in real networks, Research Report 92/2, DEC Western Research Laboratory, California, USA, April 1992.

13. J. Moy, OSPF version 2, IETF Internet RFC 2328, 1998.

14. M. J. O'Mahony, Results from the COST 239 project, ultra-high capacity optical transmission networks, Proceedings 22nd European Conference On Optical Communication 2: 11-18, 1996.

15. H. Suzuki and F. A. Tobagi, Fast bandwidth reservation scheme with multi-link and multi-path routing in ATM networks, Proceedings INFOCOM '92 11th Annual Joint Conference of the IEEE Computer and Communications Societies 3: 2233-2240, 1992.

16. C. Tang and P. K. McKinley, A distributed multipath computation framework for overlay network applications, Technical Report, Michigan State University, 2004.

17. T. Ishida, K. Ueda and T. Yakoh, Fairness and utilization in multipath network flow optimization, Proceedings 2006 IEEE International Conference on Industrial Informatics: 1096 – 1101, 2006.

18. D. Timothy, N. M. Zukerman and R. G. Addie. Modeling broadband traffic streams, Proceedings of Globecom ’99, Rio de Janeiro, Brazil: 1048 – 1052, 1999.

19. P. Traina, BGP-4 protocol analysis, IETF RFC Internet 1774, October 1995.

20. Y. Wang and Z. Wang, Explicit routing algorithms for Internet traffic engineering, Proceedings 8ht International Conference on Computer Communications and Networks: 582-588, 1999.

21. A. E. I. Widjaja, Mate: MPLS adaptive traffic engineering, Proceedings INFOCOM 2001 20th Annual Joint Conference of the IEEE Computer and Communications Societies 3: 1300-1309, 2001.

22. R. Wilder, K. Ramakrishnan and A. Mankin, Dynamics of congestion control and avoidance of two-way traffic in an OSI testbed, Computer Communication Review 21(2): 43-58, 1991.

23. L. Zhang, S. Shenker and D. Clark, Observations on the dynamics of a congestion control algorithm: the effect of two-way traffic, Proceedings of ACM SIGCOMM 1991: 133-147, 1991.

24. A. Das, C. Martel, B. Mukherjee and S. Rai, New approach to reliable multipath provisioning, IEEE/OSA Journal of Optical Communications and Networking PP (99) : 95-103, 2011.

25. B. Valery and V. Vyacheslav, The analysis of the characteristics of routing protocols in IP network, Proceedings International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET 2010): 185, 2010.

26. A. Al-Shabibi and B. Martin, MultiRoute - a congestion-aware multipath routing protocol , Proceedings International Conference on High Performance Switching and Routing (HPSR 2010): 88-93, 2010.

27. L. He, Efficient multi-path routing in wireless sensor networks, Proceedings 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM): 1-4, 2010.

28. M. Hedayati, H. R. Hoseiny, S. H. Kamali and R. Shakerian, Traffic load estimation and load balancing in multipath routing mobile ad-hoc networks, Proceedings 2nd International Conference on Mechanical and Electrical Technology (ICMET): 117-121, 2010.

29. A. Aronsky and A. Segall, A multipath routing algorithm for mobile wireless sensor networks, Proceedings 3rd Joint IFIP Wireless and Mobile Networking Conference (WMNC): 1-6, 2010.

30. Y. Chen and C. Zhang, A multipath routing protocol with path compression for ad hoc networks, Proceedings 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE) 1: 624-628, 2010.

31. A. Iselt, A. Kirstadter, A. Pardigon and T. Schwabe, Resilient routing using MPLS and ECMP, Proceedings of the Workshop on High Performance Switching and Routing (HPSR): DOI 10.1109, 2004.

32. M. Dzida, M. Zagozdzon, M. Pioro and A. Tomaszewski, Optimization of the shortest-path routing with Equal-Cost Multi-Path load balancing, Proceedings of the International Conference on Transparent Optical Networks 3: 9-12, 2006.

CALL FOR PAPERS

Journal: International Journal of Computer Networks (IJCN)
Volume: 3 **Issue:** 1
ISSN:1985-4129
URL: http://www.cscjournals.org/csc/description.php?JCode=IJCN

About IJCN

The International Journal of Computer Networks (IJCN) is an archival, bimonthly journal committed to the timely publications of peer-reviewed and original papers that advance the state-of-the-art and practical applications of computer networks. It provides a publication vehicle for complete coverage of all topics of interest to network professionals and brings to its readers the latest and most important findings in computer networks.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCN.

IJCN List of Topics

The realm of International Journal of Computer Networks (IJCN) extends, but not limited, to the following:

- Algorithms, Systems and Applications
- ATM Networks
- Cellular Networks
- Congestion and Flow Control
- Delay Tolerant Networks
- Information Theory
- Metropolitan Area Networks
- Mobile Computing
- Multicast and Broadcast Networks
- Network Architectures and Protocols
- Network Modeling and Performance Analysis Network
- Network Security and Privacy
- Optical Networks
- Personal Area Networks
- Telecommunication Networks
- Ubiquitous Computing
- Wide Area Networks
- Wireless Mesh Networks
- Ad-hoc Wireless Networks
- Body Sensor Networks
- Cognitive Radio Networks
- Cooperative Networks
- Fault Tolerant Networks
- Local Area Networks
- MIMO Networks
- Mobile Satellite Networks
- Multimedia Networks
- Network Coding
- Network Operation and Management
- Network Services and Applications
- Peer-to-Peer Networks
- Switching and Routing
- Trust Worth Computing
- Web-based Services
- Wireless Local Area Networks
- Wireless Sensor Networks

IMPORTANT DATES

Volume: 3
Issue: 1
Paper Submission: January 31, 2011
Author Notification: March 01, 2011
Issue Publication: March /April 2011

CALL FOR EDITORS/REVIEWERS

CSC Journals is in process of appointing Editorial Board Members for ***International Journal of Computer Network (IJCN)***. CSC Journals would like to invite interested candidates to join **IJCN** network of professionals/researchers for the positions of Editor-in-Chief, Associate Editor-in-Chief, Editorial Board Members and Reviewers.

The invitation encourages interested professionals to contribute into CSC research network by joining as a part of editorial board members and reviewers for scientific peer-reviewed journals. All journals use an online, electronic submission process. The Editor is responsible for the timely and substantive output of the journal, including the solicitation of manuscripts, supervision of the peer review process and the final selection of articles for publication. Responsibilities also include implementing the journal's editorial policies, maintaining high professional standards for published content, ensuring the integrity of the journal, guiding manuscripts through the review process, overseeing revisions, and planning special issues along with the editorial team.

A complete list of journals can be found at http://www.cscjournals.org/csc/byjournal.php. Interested candidates may apply for the following positions through http://www.cscjournals.org/csc/login.php.

Please remember that it is through the effort of volunteers such as yourself that CSC Journals continues to grow and flourish. Your help with reviewing the issues written by prospective authors would be very much appreciated.

Feel free to contact us at coordinator@cscjournals.org if you have any queries.

Contact Information

Computer Science Journals Sdn BhD
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: +603 6207 1607
+603 2782 6991
Fax: +603 6207 1697

BRANCH OFFICE 1
Suite 5.04 Level 5, 365 Little Collins Street,
MELBOURNE 3000, Victoria, AUSTRALIA

Fax: +613 8677 1132

EMAIL SUPPORT
Head CSC Press: coordinator@cscjournals.org
CSC Press: cscpress@cscjournals.org
Info: info@cscjournals.org

COMPUTER SCIENCE JOURNALS SDN BHD

M-3-19, PLAZA DAMAS

SRI HARTAMAS

50480, KUALA LUMPUR

MALAYSIA