A New Implementation for Measuring Delay and Radio Frequency/Ability in Wireless Networks

Sanyasi Naidu K[#1], Santosh Naidu P[#2]

[#1,2]Computer Science and engineering
[#1,2]MVGR College of Engineering, Vizianagaram, Andhra Pradesh, India
[1]ksnaidu333@gmail.com
[2]amsan2015@gmail.com

Abstract— **Normally delay in the previous work on only on mesh topology, but on wireless sensor networks but always experimentally calculation and rectification of delay in the Wireless Sensor Network is a big challenge. Once the Wireless Sensor Network created statically or energetically/changing quickly as needed router or central server have to face big very hard to maintain the peers information. We use grouped flows (machine/method/way) with (making two or more things look the same or happen at the same time) to get the perfect delay in the session wise. Once the path(s) established for transmission the source and destinations will be under one session for logging (further checking (for truth)). The moment the data transmission is started our approach will check the session for any existing session which is matching with the current transmission. If it matches, the communication will not get copied and ignored and also log will not be updated. If any session is not matching with current transmission session then this session finds the shortest path for transmission and also calculates the average delay for that session's all communications.**

Keywords— **SNMP, LDA, Switched Ethernet Network.**

I. Introduction

We choose wireless (something made for a particular reason) network for the average delay calculations for session's broad casting. Session, normally whenever the transmission path(s) established for communications one session is established and the whole information will be saved in log for avoiding copy and for grouped flows which was not there in the previous work.

In this paper we had a new approach called session builder. This approach is handy for Wireless Sensor Network (basic equipment needed for a business or society to operate) for finding out the sessions that removes the need of trace backing (one limit/guideline in this case is avoiding retransmission). This in turn helps us by providing security so that each time we go for a new selection node the need of finding out for (able to be done) paths will be reduced. This set of computer instructions is also useful for deciding/figuring out whether a particular node is a router.

In this work we also had another approach for router monitoring. The main aspect behind this set of computer instructions is that it checks for the copies of sessions. While transmitting data it checks for the packets that are being transmitted. If same data is being transmitted it nullifies the path and uses the session that has been before now build. This approach is known as cloning or casting. If the data is not almost the same as the previous one then it will be automatically updated in the log table.

Practical example: let us take to set of grouped flows. Source as p1 and destinations are p2{, p6}, source as p2 and destinations p3{, p4} and Central monitoring system will start session for first grouped flow. Once the session is established session builder will start checking in the past log for any available session. And if that session is matching then session builder will reuse that session till grouped flow and it will calculate the average delay and update in the log. Session builder will calculate the time of transmission for all sessions. And will be updated in the session's log.

For our first set of communication the source as P1 and destinations as P2 and P6, now session builder will establish the paths. In this case the available paths for *transmissions are p1-p3 - p6, p2-p3, p2-p4-p3. Session builder establishes a session for communication and checks for the log if any matching sessions for the above paths. If system finds any existing or matched log the session will be reused for (happening at the same time/having the same exact contents) broadcasting and delivers the packets from p1 to p6 by finding the shortest path and update the time of communication, average delay for all the session's path. Finally calculate the average delay and marks for the (dividing line/point where something begins or changes) value. If average delay is less than the (dividing line/point where something begins or changes) value then that session will marked for avoiding the session track.

II. Literature Survey

An increasing number of datacentre network applications, including automated trading and high-performance figuring out/calculating, have strict end-to-end delay needed things where even microsecond differences/different versions may be terrible (and impossible to put up with). The resulting fine-grained measurement demands cannot be met effectively by existing technologies, such as SNMP, Net Flow, or active probing. We propose instrumenting routers with a hash-based (very simple/from a time very long ago) that we call a Lossy Difference Aggregator (LDA) to measure delays down to tens of microseconds even in the presence of packet loss. Because LDA does not change or combine all the features of the packet, it can be sent out and used (in little steps) without changes along the forwarding path. When compared to Poisson-spaced active probing with almost the same overheads, our LDA (machine/method/way) delivers (many, many times more/much, much less) smaller relative error; active probing needs/demands 50-60 times as much radio frequency/ability to deliver almost the same levels of (quality of being very close to the truth or true number). Although (existing everywhere) use/military service is (in the end) desired, it may be hard to (accomplish or gain with effort) in the shorter term; we discuss a partial use/military service (related to the beautiful design and construction of buildings, etc.) called mPlane using LDAs for intra-router measurements and localized part/section measurements for inter-router measurements.

Traffic measurement is a critical part for the control and engineering of communication networks. We argue that traffic measurement should make it possible to get the spatial flow of traffic through the domain, (in other words), the paths followed by packets between any (entry/going in to something) and exit point of the domain. Most useful thing/valuable supply setting apart and distributing and ability (to hold or do something) planning tasks can benefit from such information. Also, traffic measurements should be received/be gotten without a routing model and without knowledge of network state. This allows the traffic measurement process to be tough to network failures and state doubt. We propose a method that allows the direct guessing (based on what you've been told) of traffic flows through a domain by watching/following the paths of a subset of all packets going through the network. The key advantages of the method are that (i) it does not depend on routing state, (ii) its putting into use cost is small, and (iii) the measurement reporting traffic is modest and can be controlled exactly. The key idea of the method is to sample packets based on a hash function figured out/calculated over the packet content. Using the same hash function will produce the same sample set of packets in the entire domain, and enables us to reconstruct packet paths.

In this paper we present new modelling and analysis way of doing things for describing the steady state performance of a TCP flow. We invert the loss process in our model, by treating loss events arriving at the source as a Poisson stream rather than packets going out on the network with some loss probability p. This enables us to model the window size behaviour as a Poisson Counter driven (random/including random data points) Differential Equation and perform analysis. We use the data collected in [1] to validate our modelling and analysis way of doing things. Results point to/show that our model can capture the behaviour of TCP throughput quite (in a way that's close to the truth or true number). Our way of doing things enables simple fluid analysis of TCP and other TCP-like crowding and blockage control (machine/method/way).

Wall Street's search for speed is not only putting floor traders out of work but also opening up space for new other choice exchanges and electronic communications networks that fight against the established stock markets. Electronic trading has reduced overall dangerous nature/wild up and down prices in the equities markets, because dangerous nature/wild up and down prices is a product of group of animals buying or selling, and electronic trading--responding immediately to tiny price ups and downs--tends to smooth out such mass behaviour. And it has given established exchanges with new money/money income opportunities, such as co-location services for companies that wish to place their servers in direct physical closeness to the exchanges' systems. Electronic trading also has created opportunities for a new class of vendors--execution services firms and systems integrators promising the fastest possible transaction times.

III. Existing System

An increasing number of data centre networks applications and automated trading having strict end to end delay needed things where even micro second difference/different version even terrible (and impossible to put up with).

1. Existing systems are SNMP, net flow cannot met effectively
2. In intranet network the router maintains the LDA set of computer instructions to find out the fine grained delay
3. This approach is not related for wireless Ad Hoc networks.

Limitations

1. Normally delay in the previous works on only on mess topology, but on wireless sensor networks but always experimentally calculation and rectification of delay in the WSN.
2. As the WSN created statically or energetically/changing quickly as needed, the router or central servers have to face big very hard to maintain the peers information.

IV. Proposed System

The moment the data transmission is started our approach will check the session for any existing session which is matching with the current transmission.

1. Session builder is an approach for finding the sessions in the transmission. For each transmission one session will created
2. This session stores the data about source, destination and the router which routes data and the calculated delay by the router.

Features

1. Approach is handy for WSN (basic equipment needed for a business or society to operate) for finding out the sessions that removes the need of trace backing (one limit/guideline in this case is avoiding retransmission).
2. This set of computer instructions is also useful for deciding/figuring out whether a particular node is a router. So that the data security is promised to/certain by not moving (from one place to another) the packets of data to that particular node if it is a router.

Algorithm (Network analyzation on controlling and optimization)

This network can analyse on controlling and optimization (NACO) is the technique to calculate aggregate latency based on growing sessions. Once the network is deployed for first time there will not be any log initially. The sessions will be logged and updated in the log table with the following sequence.

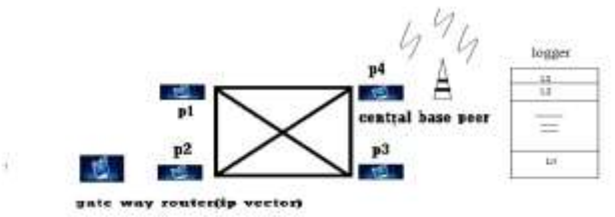


Figure 1 Gateway Router

Router Node	Source Node	Destination Node	Session Id	Latency	File size	Bandwidth
Node 1	N5	N6	104	1406	376	0.0341394
Node 0	N2	N4	13	1594	262	0.0338770

Table 1 Log table

The main usage on this log table is to enforce the energetic/changing router/session for putting central storage place router which will monitor whole transmission of the casting nodes (destinations) from source node which is selected and controlled by central storage place node. The above table shows the experimental result for delay calculation on created radio frequency/ability.

Dynamic generation of router/session: If we have the nodes A{, B, C, D, E} which are peers (in little steps) maintained under session builder (with log updating). It the sessions keeps growing for transmission the session tracker will keep watching and following the success transmission for that particular transmission (si to dj) will be watched and followed for further session (id) comparison to assign energetic/changing router assignment. So once the session is updated in the log table with animal desires that route will be (figured out the worth, amount, or quality of) based on source and destinations packet comparison. The nodes preauthorized according to success rate of transmission in decreasing model.

Duffield and M.Grossglauser have proposed a method for the consistent sampling of packet paths in a network. The sampling selects a subset of packets, but if a packet is selected at one link, it will be selected at every other link it goes through. Ongoing through the network, each packet completely/in a hinting way points to/shows whether or not it should be sampled through its invariant part, (in other words), those bits that do not change from link to link. A hash of these bits it calculated at each router, and only those packets whose sampling hashes fall within a given range of values are selected. For selected packets, a different hash, the identification hash, is used to stamp an identity on the packet. This is communicated by the sampling router to the measurement systems. This enables post sampling analysis of clear/separate paths once the samples are reported.

N. Duffield, A. Gerber, and M. Gross glauser reported on the Path Engine, and early model back-end system to receive path samples from network elements, reconstruct and store packet paths in a (computer file full of information), and furnish questions through a GUI. The goal was to demonstrate how such a system could provide network operators with new disease-identifying tools not available with current network measurements. The main technical challenges rose up from the effects of label crashes in samples, the possibly huge amounts of raw data involved, and the need to find an acceptable agreement (where everyone meets in the middle) between question (statement that's not detailed) on the one hand, and data volumes and question speed on the other. The reasons (for doing something) for the design choices that we employed to meet these challenges make up/be equal to the main work reported here. These were: how sampling limits/guidelines could be chosen to control both sample volumes, and the frequency of label crashes; a timer-based (machine/method/way) for the reconstruction of paths; and the systems and setup employed for data management and questioning.[2]

N. Duffield has argued that when network link performance characteristics can be well separated into two categories, good and bad, a simple guessing-related set of computer instructions--that attributes path failure to the smallest set of regular, dependable connection failures--can be effective in identifying candidate bad links on a tree from end-to-end measurements. This approach is (done for good reason) by the observation that when bad links are uncommon, the two or more badly performing paths likely have a bad link in their intersection. More than that, the likelihood for this to happen is relatively insensitive to changes if the fraction of (uncommon) bad links. (looking at things in the opposite way), the false positive rate is very low in this government in power, because only those (rare) good links at the head of maximal bad sub trees are falsely thought of/considered bad [2].

A.Montanari, B.Prabhakar, S.Dharmapurikar, and A.Kabbani presented Counter Braids, a efficient minimum-space counter (related to the beautiful design and construction of buildings, etc.), that solves large-scale network measurement problems such as per-flow and per-prefix counting. Counter Braids (in little steps) compresses the flow sizes as it counts and the message passing reconstruction set of computer instructions recovers flow sizes almost perfectly. We (make something as small as possible/treat something important as unimportant) counter space with (in small steps up) compression, and solve the flow-to-counter association problem using random graphs. As shown from real trace test runs (that appear or feel close to the real thing), we can count upto 1 million flows purely in SRAM and recover the exact flow sizes. We are now putting into use this in an FPGA to decide/figure out the actual memory usage and to better understand putting into use issues [3].

V. Theoretical Analysis and Methods

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node/peer will be with several parts: a radio transceiver with an inbuilt antenna or connection to an attached externally built antenna, an electronic controller, and digital circuit for interfacing with the sensors and the resource of energy, usually a battery or an embedded form of energy consuming.

A sensor node/peer might differ in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created.

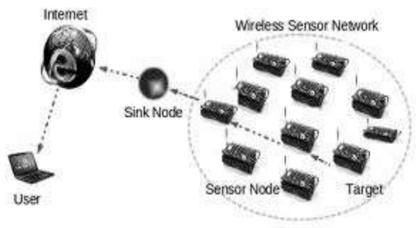


Figure 2 Wireless Sensor Network

Algorithm Description

Firstly we consider a vector of source nodes. We also consider a destination node. Combined we pass them to the available path function. This function returns the nodes that are intermediate to the source and destination. In this function we also check if the nodes are routers or not. If it returns true then we make the packet size invisible to that particular router. If it returns false we build a new session using the intermediate node and destination. Another algorithm is for router monitoring. This will check the data that is being transmitted. If same data packets are found we simply use the old session that has been built previously. Else we update the info in log. This concept is known as cloning.

Example

Let us assume a source nod from the above figure as node ‘A’ also destination ‘D’. Now passing these two nodes to the available path function that will return the paths available, let us suppose that to be ‘A-B-D’. In the function itself another task will be done i.e. checking router. If ‘B’ is determined as router then packets are nullified to that particular router. In this work session will take important role to calculate the best latency in through put with session builder approach. The average flows are always depends on the various factors in sensor networks. In this we take mesh topology for sensor network. The reason behind the tree based sensor network is to have shortest path in short time. The shortest path discovery is in less time in wireless sensor network. In this approach we put some marker at the transmission side and log will be updated and once the shortest path found which will leads for broad casting. The broadcasting will be logged for the same path. For the same path depends on % of transmission average latency will be calculated with our session builder approach.

Here the router will be monitoring the log table continuously for the loss of aggregated flows and controls and ignores the regular over all latency paths. These paths will be totally vanished for better broad casting with low level latency. The main advantage for this approach is to adopt the current network to other Ad hoc and regular topologies. Overall latency measurements are globalized to single one for various factors like throughput, bandwidth variation, and large data transmissions. Once the network is deployed for first time there will not be any log initially. The sessions will be logged and updated in the log table with the following sequence.

Table 2 Log table

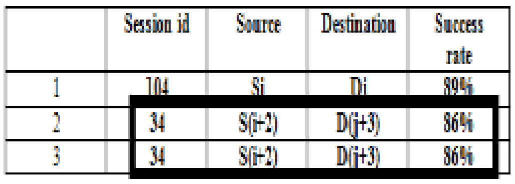
Router Node	Source Node	Destination Node	Session Id	Latency	File size	Bandwidth
Node 1	N5	N6	104	1406	376	0.0341394
Node 0	N2	N4	13	1594	262	0.0338770

The main usage on this log table is to enforce the dynamic router/session for putting central repository router which will monitor whole transmission of the casting nodes (destinations) from source node which is selected and controlled by central repository node. The above table shows the experimental result for latency calculation on generated bandwidth.

Dynamic Generation of Router/Session

If we have the nodes {A, B, C, D, E} which are peers incrementally maintained under session builder (with log updation). It the sessions keeps growing for transmission the session tracker will keep tracking the success transmission for that particular transmission (si to dj) will be tracked for further session (id) comparison to assign dynamic router assignment. So once the session is updated in the log table with id that route will be evaluated based on source and destinations packet comparison. The nodes preauthorized according to success rate of transmission in decreasing model.

Table 3 Path Cloning



	Session id	Source	Destination	Success rate
1	104	Si	Di	89%
2	34	S(i+2)	D(j+3)	86%
3	34	S(i+2)	D(j+3)	86%

In the above case same session id’s with same transmission and in this case if the transmission based on the same packets the session will be established but in fact this will be ignored for further transmission. The reason behind this is if all the sessions are unique (without transmission paths) will be considered but with transmission paths are considered for ignorance of transmission. The generated sequence resembles on the

```
//Initialization
Source= Vsrcε {v1, v2, v3…., vn}
Destination= Vdes ε {v1, v2, v3…., vn}
Ti= start time
Tf= final time
//Process
Start:
ti@Vsrc
tf@Vdes
Latency = tf-ti
End;
```

The Aggregated Latency calculation

The above the table contains the latency for 2 sessions. But the all session's aggregated latency calculation is as follows. If n is total number of sessions (ignored transmission). All latencies exist in Li so the latencies aggregation is depend on total number of sessions.

I = n-m and $\int_{0}^{i-1}$ (n-m)/n(math part for aggregation) // is the total aggregated latency.

This serves to explain the sources of latency on a switched Ethernet network and describe how to calculate cumulative latency as well as provide some real world examples. Latency in a communications network is defined as the time it takes for a message to traverse the network from the transmitter to the receiver. In certain applications, like voice or real-time automation, the network must guarantee a certain maximum latency or the application may not work in a satisfactory manner or, worse, may fail outright. Switched Ethernet networks have several sources of latency: 1) store and forward, 2) switch fabric processing, 3) wire line transmission, and 4) frame queuing. All of these latencies except for queuing are deterministic and yet the effects of frame queuing can also be calculated providing one knows the nature of all sources of traffic on the network.

Sources of Latency

Store and Forward Latency (LSF)

Store and forward refers to the basic operating principle of an Ethernet switch. The term is descriptive of its actual operation: the switch stores the received data in memory until the entire frame is received. The switch then transmits the data frame out the appropriate port(s). The latency this introduces is proportional to the size of the frame being transmitted and inversely proportional to the bit rate as follows:

LSF= FS / BR

Where LSF is the store and forward latency, FS is the frame size in bits, and BR is the bit rate in bits/s. For the maximum size Ethernet frame (1500 bytes) at 100 Mbps the latency is 120μs. For comparison, the minimum size frame (64 bytes) at Gigabit speeds has a latency of just 0.5μs.

Wire line Latency (L_{WL})

Bits transmitted along a fiber optic link travel at about ⅔ of the speed of light (3x108 m/s) When very long distance Ethernet links are deployed, this delay can become significant. The one way latency for a 100km link works out to:

L_{WL} = 1x105 m / (0.67 × 3×108 m/s) ≈ 500 μs

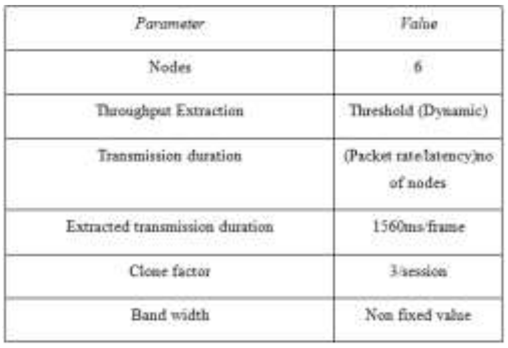
Note that for the distances involved in local area networks, this delay becomes trivial compared with the other contributions to latency.

VI. Experimental Analysis

Our proposed algorithm has been developed on a simulator using Java Beans. Wireless ad hoc ne Networks with session, and central router is designed. Transmission links are used to connect these nodes. The Tree structure and the session table of transmitted paths to users are displayed, once the logging is done for that run. Average Throughput is calculated in Mbps at transmission.

To simulate the model, the following simulation parameters are considered:

Table 4 Simulation Parameters



Parameter	Value
Nodes	6
Throughput Extraction	Threshold (Dynamic)
Transmission duration	(Packet rate latency)no of nodes
Extracted transmission duration	1560ms/frame
Clone factor	3/session
Band width	Non fixed value

Implementation Methodology and Results

In this copying /cloning of session reuse method is used

Parameter includes

1. Session id host or peer id
2. Rate of transmission
3. Packet frame (size)
4.

Rate of transmission

- ➢ The effective transmission rate is calculated as (a) the measured number of units of data, such as bits, characters, blocks, or frames, transmitted during a significant measurement time interval divided by (b) the measurement time interval.
- ➢ The effective transmission rate is usually expressed as a number of units of data per unit time, such as bits per second or characters per second

The Input Parameters should be taken from each user.

1. Source node.
2. Destination id
3. Session id.
4. The data rates each transmission.
5. Packet size

Since we have hundreds of users in real time scenario, we took these parameters randomly and 1 - UGS (Unsolicited Grant Services).

2 - rtPS (Real-Time Polling Service).
3 - nrtPS (Non Real-Time Polling Service).
4 - Best Effort (BE).

UGS and rtPS are real time applications and require constant bit rate traffic to be serviced and are no way delay tolerant for which are given highest priority than nrtPS and BE which are variable bit rate and delay resistant. After taking these parameters and precautions into account, we designed different Multihop tree structures after each run of the algorithm are as follows:

The below tree structure depicts the connectivity between nodes and router, in wireless adhoc network the nodes dynamically, and one router is generated. This router is manages the network and maintains the log table for transmissions .for each and every transmission the source and destination nodes are asked by the admin and router will manage the nodes. This router changes randomly in the network.

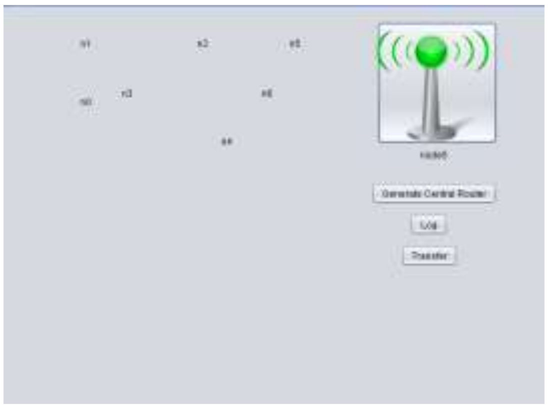
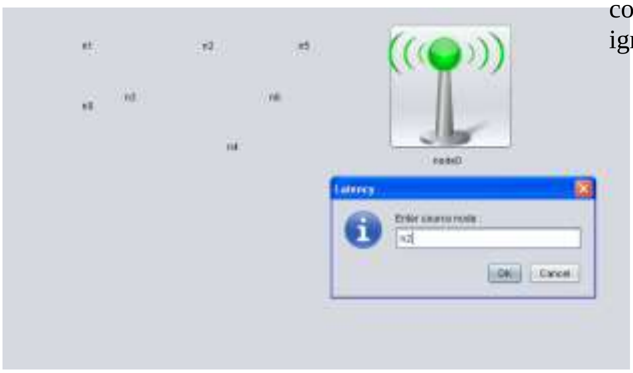


Figure 3 Ad-hoc tree structure

The session Algorithm is implemented and the final allocation of tiles is tabulated with columns of:

1. Segment allocated.
2. Number of tiles allocated (each tile is having fixed data rate of 1Mbps).
3. Number of tiles remaining (each tile is having fixed data rate of 1Mbps).
4. Delay in Allocation (is scaled to 0.1 ms).



Enter source node :

Figure 4 Selection of nodes

The above scenario is done for ‘6’ number of tree structures until no requests from. In the network select on source node randomly and select destination node for transmit the data. This nodes information update in the log table by the router.

Figure 5 File Selection

Select file for transmitting in the network. File name, file size is updated in the log table.

Figure 6 session id creation

Once selecting the source and destination and the file for transmission then one session id is created for that file transmission by the router. This session id is stored in the log table Based on this session id the source and destination the router checks the redundant paths. This transmission simply ignores the total path and the log table cant updated the duplicated path details session id.

Here the router will be monitoring the log table continuously for the loss of aggregated flows and controls and ignores the regular over all latency paths.

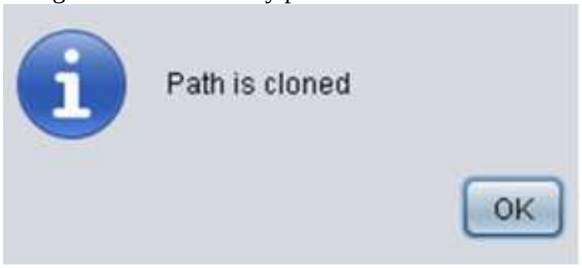
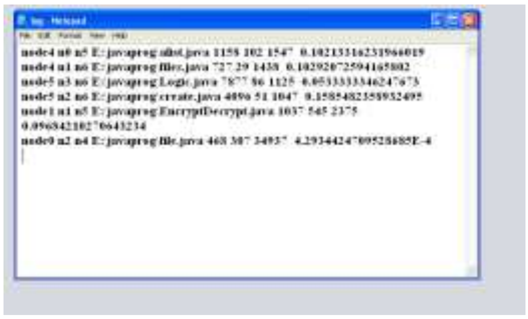

Path is cloned

Figure 7 path is cloned

In the transmission the if the same source and destinations are occurred then that path is cloned because the same data transmission is not necessary send again. It reduces the latency through over all transmission

Same session id’s with same transmission and in this case if the transmission based on the same packets the session will be established but in fact this will be ignored for further transmission. The reason behind this is if all the sessions are unique (without transmission paths) will be considered but with transmission paths are considered for ignorance of transmission.



```
node4 n0 n5 E:javaprog alist.java 1158 102 1547  0.10213316231966019
node4 n1 n6 E:javaprog files.java 727 29 1438  0.1029207259416580 2
node5 n3 n6 E:javaprog Logic.java 7877 86 1125  0.0533333346247673
node5 n2 n6 E:javaprog create.java 4096 51 1047  0.1585482358932495
node1 n1 n5 E:javaprog EncryptDecrypt.java 1037 545 2375
0.09684210270643234
node0 n2 n4 E:javaprog file.java 468 307 34937  4.293442470952868 5E-4
```

Figure 8 Calculated Latency

VII. Results

Delay

The above table shows consolidated experimental results for each session. The session id is not shown with casting duplications in this table. These results are experimentally proven for aggregated latency / session (math part for aggregation).

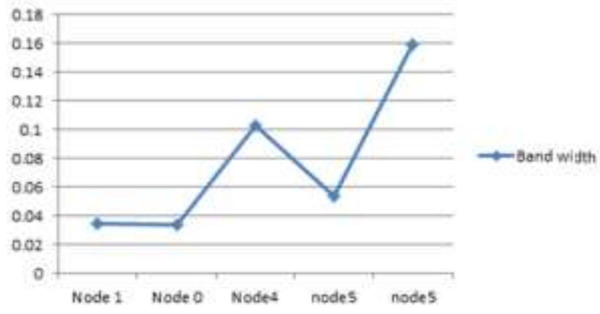


Figure 9 Bandwidth graph

Ex: let’s take the morphed set of the experimental set. R = 0.10292072

Lets the value (first factor) is the estimated value for simulation it t1 and t2 is bandwidth and latency / session is as follows.

Latency = t1/t2

And total aggregated latency will be as follows first need to remove the casting nodes sessions from consolidations(m) from total sessions n-m and aggregation is n-m/n which is aggregated latency and this can be extended for further network adaptation which is aggregated latency/network(f+1) with f as basic existing network.

```
node4 n0 n5 E:javaprog alist.java 1158 102 1547  0.10213316231966019
node4 n1 n6 E:javaprog files.java 727 29 1438  0.1029207259416580 2
node5 n3 n6 E:javaprog Logic.java 7877 86 1125  0.0533333346247673
node5 n2 n6 E:javaprog create.java 4096 51 1047  0.1585482358932495
node1 n1 n5 E:javaprog EncryptDecrypt.java 1037 545 2375
0.09684210270643234
node0 n2 n4 E:javaprog file.java 468 307 34937  4.293442470952868 5E-4
```

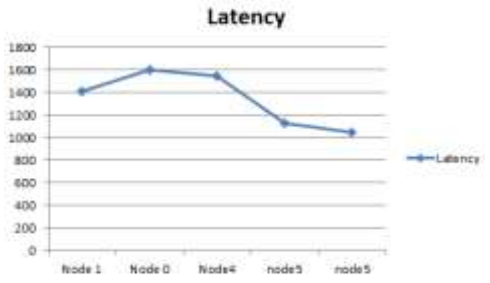


Figure 10 Graph showing latency measurement

VIII. Conclusion

In this work session will take important role to calculate the best delay in through put with SESSION BUILDER approach. The average flows are always depends on the different factors in sensor networks. In this we take tree based sensor network. The reason behind the tree based sensor network is to have shortest path in short time. The shortest path discovery is in less time in wireless sensor network. In this approach we put some marker at the transmission side and log will be updated and once the shortest path found which will leads for broad casting. The broadcasting will be logged for the same path. For the same path depends on % of transmission average delay will be calculated with our session builder approach.

Here the router will be monitoring the log table continuously for the loss of grouped flows and controls and ignores the regular over all delay paths. These paths will be totally disappeared for better broad casting with low level delay. The main advantage for this approach is to put into use the current network to other ad-hoc and regular topologies. Overall delay measurements are globalized to single one for different factors like throughput, radio frequency/ability difference/different version, large data transmissions.

IX. Future Work

This work can be extended to calculate grouped delay for adoptive networks; the networks can be adopted with the current network who is in progress with grouped delay calculation frame work. The adoptive feature for the current network is always in the view of (the total of something over time) delay calculation using gateway ip/adoptive network. In this case gateway IP is main source for the third party network's adoption.

Acknowledgment

I would like to thank Department of Computer Science and Engineering staff and management of MVGR College of Engineering for their support in doing this work.

References

[1] Cisco, "Cisco extends highly secure, improved communications in rugged environments with new family of industrial Ethernet switches," 2007 [Online]. Available: http://newsroom.cisco.com/dlls/ 2007/prod_111407.html

[2].DuffieldandM.Grossglauser,"Trajectorysamplingfordirectt raffic observation," in Proc. ACM SIGCOMM, Aug. 2000, pp. 271–282.

[3] N. Duffield, A. Gerber, and M. Grossglauser, "Trajectory engine: A backend for trajectory sampling," in Pro. IEEE Netw. Oper. Manage. Symp., Apr. 2002, pp. 437–450.

[3] N. Duffield, "Simple network performance tomography," in Proc. USENIX/ACM Internet Meas. Conf., Oct. 2003, pp. 210–215.

[4]Y.Lu,A.Montanari,B.Prabhakar,S.Dharmapurikar,andA.Ka bbani, "Counter braids: A novel counter architecture for per-flow measure- ment," in Proc. ACM SIGMETRICS, Jun. 2008, pp. 121–132.

[5] S. Machiraju and D.Veitch, "A measurement-friendly network(MFN) architecture," in Proc. ACM SIGCOMM Workshop Internet Netw. Manage., Sep. 2006, pp. 53–58. [6] J.Mahdavi,V.Paxson,A.Adams,andM.Mathis,"Creatingascalab le architecture for Internet measurement," in Proc. INET, Jul. 1998.

[7] R. Martin, "Wall street's quest to process data at the speed of light," InformationWeek 2007 [Online]. Available: http:// www.informationweek.com/news/infrastructure/showArticle. jhtml?articleID=199200297

[8] V. Misra, W.-B. Gong, and D. Towsley, "Stochastic differential equa- tionmodeling and analysis of TCP windowsizebehavior," in Proc. IFIP WG 7.3 Perform., Nov. 1999.

[9] M. Riska, D. Malik, and A. Kessler, "Trading flow architec- ture," Cisco Systems, Inc., San Jose, CA, Tech. Rep., 2008 [Online]. Available: http://www.cisco.com/en/US/docs/solutions/ Verticals/Trading_Floor_Architecture-E.pdf

[10] T. Szigeti and C. Hattingh, "Quality of service design overview," Cisco, San Jose, CA, Dec. 2004 [Online]. Available: http:// www.ciscopress.com/articles/article.asp?p=357102&seqNum =2