INLyD: Inter-Network-Layer Delay as a Low-cost Quality Metric for Multi-hop Routing in Wireless Mobile Networks

Syed Rehan Afzal
Eindhoven University of Technology, The Netherlands
s.r.afzal@tue.nl

Majid Nabi
Eindhoven University of Technology, The Netherlands
m.nabi@tue.nl

Sander Stuijk
Eindhoven University of Technology, The Netherlands
s.stuijk@tue.nl

Twan Basten
Eindhoven University of Technology and TNO-ESI, The Netherlands
a.a.basten@tue.nl

ABSTRACT
The need for authentic and effective portrayal of the spatio-temporally changing quality of wireless links has gained wide attention especially over the last decade. Software-based link quality estimators (LQE) classify links with help of packet reception ratio (PRR), required number of packet transmissions (RNP) and scoring/grading schemes that again utilize PRR, RNP or retransmission based heuristics. On the contrary, this paper makes a case for inter-network-layer delay as a classification metric to boost end-to-end packet delivery in multi-hop communication. In essence our Inter-Network-Layer Delay metric (INLyD) uses a simplistic receiver-side in-band signaling scheme to passively accumulate queuing, retrying, back-off, transmission and propagation delay statistics while generating no additional control packet overhead. Our experiments show that the INLyD metric is not only light-weight (25% less MAC transmissions required per node) but substantially outperforms proactive broadcast based estimation schemes in static and mobile scenarios (1.7 and 1.2 times more end-to-end UDP delivery respectively for the performed experiments).

General Terms
Design, Experimentation, Measurement, Performance

Keywords
Asymmetric link quality; passive link quality measurement; wireless ad hoc networks; wireless sensor networks

1. INTRODUCTION
Floods in 2015 Southern India left millions of people displaced with telecom and electricity services disrupted for several days [10]. Improving network device life span in domains such as Mobile ad hoc networks (MANETs) for disaster relief camps or festivals etc. or long term deployed Wireless Sensor Networks (WSNs) is paramount. Life time is directly proportional to energy consumption. Energy consumed to transmit a single bit over the wireless channel is shown equivalent to that consumed to carry out approximately 400 [15] to 800 [2] instructions (depending on the sensor node used). Quality-of-Service (QoS) provisioning schemes are employed to reduce packet losses. The goal of network QoS provisioning is to achieve a more deterministic behavior of the network apparatus involved. In a lossy, unreliable and tempro-spatially changing medium such as wireless, the need for effective QoS estimation becomes imperative.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
PE-WASUN'16, November 13-17, 2016, Malta, Malta
© 2016 ACM. ISBN 978-1-4503-4505-7/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2989293.2989307

Since QoS subscription always comes with its costs, the effectiveness of a given QoS metric is a subjective term and depends on the network and respective application requirements. Furthermore, these qualities are almost always interdependent; e.g., improving device coverage by increasing transmission power will result in reduced device lifetime. The provision of QoS in resource-scarce networks such as WSNs and MANETs introduces additional challenges to the task. An effectiveness of a QoS metric or routing scheme in this context therefore is to efficiently utilize the network resources by offering improved performance in terms of delivery ratio, bandwidth, latency and throughput at the expense of as little as possible overhead. Several Link Quality Estimators (LQEs) have been proposed over the years where throughput and Packet Reception Ratio (PRR) have gained special interest [4], [5], [6], [7] and [8].

There are two approaches to measure link quality namely active link monitoring and passive link monitoring. Active link monitoring requires each device to proactively transmit periodic broadcast or unicast control packets. The link quality is then estimated from the measured Quality of Experience (QoE) of these control packets. Active link monitoring has its benefits. However, from our previous work [7], we have demonstrated the following observations:

1) Impact of active link monitoring on traffic: The overhead resulting from proactive control packet based estimation may outweigh the benefits in certain network conditions. Proactive control packets affect the nodes and the data. From a node's perspective the energy spent in physically transmitting these proactive control packets reduces the device's life span. Collisions between proactive control packets and native data packets result in more transmissions per packet delivery thereby significantly reducing the data throughput.

2) Dissimilitude of nature between control and data packets: The main advantage of an active link monitoring scheme is its characteristic ever-available quality statistics even for the links that are in-accessible or are active and accessible but have not yet been delegated any traffic stream. The aforementioned is achieved due to the fact that in such schemes unused links compute link quality just the same way as in-use links i.e. by proactively sending control packets all the times and estimating the channel response to them. This in essence means that such schemes use the QoE of these proactive control packets in the network as a predictor of how actual data packets will be treated. We have shown that this approach has its drawbacks, mainly due to the fact that the control packets cannot accurately represent the data packets [7]. This is mainly because in most application environments unicast packets have dissimilar packet size, data rate and PHY modulation when compared to broadcast packets. Unlike regular data packets, broadcast packets do not benefit from

the MAC layer RTS/CTS mechanism as well as MAC retransmissions. In other words, not only do such active link monitoring schemes inadequately represent data packets; the extra packets generated for measurement cause additional interference and collisions with the native data packets. Dissimilitude of nature between actual data packets and control packets is therefore the Achilles heel of active LQEs. The overhead of control packets caused by such schemes makes the over-all impact even worse.

In this paper we propose Inter-Network-Layer Delay (*INLyD*) which - just as inter-node-delay means delay between two nodes - implies delay between two network layers of two distinct in-range nodes. Based on our observations about deployment environments, and effectiveness and implementation practicality of link quality metrics, we have set forward *three design goals* for our metric. First, the metric must solely work on natively generated traffic and not introduce any additional control packets in the network. Second, the metric must not require any modifications in PHY, MAC or application layers. Third, the metric must be independent of any routing protocol.

Up until now research on delay based metrics have mostly been confined to channel selection protocols [20] or for timeliness related real-time or multimedia applications [21] [22]. However, in this paper we show that an inter-network-layer delay accounts for several PHY, MAC and network layer QoS related statistics. Passively monitoring data packets traffic has been shown to be more accurate in link estimation than active monitoring control packets [17]. It is however argued that such schemes usually rely on collecting statistics via packet over-hearing or altering device sleep cycle - which in return consumes significant energy [23] [24]. INLyD on the other hand only evaluates the active in-range nodes. For a given packet transmission, INLyD effectively measures seven delays namely processing delay, queuing delay, carrier sense delay, back-off delay, transmission delay, retransmission delay and propagation delay between the communicating nodes. More importantly though, INLyD relies solely on natively generated data and control packets to assess quality, and does not generate any additional control packets. Our results show INLyD yielding higher end-to-end packet delivery in most of the experimental setups, that include static, mobile, low traffic, high traffic, and uniform as well as varied traffic scenarios. The impact of accurate estimation with no extra control is shown with INLyD resulting in lower collisions and loss rate which directly translates to increased battery and network lifetime. Furthermore, in terms of costs, INLyD in comparison expends the least number of MAC transmissions per successful end-to-end UDP delivery thereby yielding the highest end-to-end goodput in most of the experimental setups.

The rest of the article is organized as follows. We discuss the different LQE metrics and estimation approaches as our related work in Section 2. Section 3 formulates the generic network attributes common between quality estimation in wireless network communication in general. Section 4 presents the detailed description and functioning of our proposed metric, INLyD. Section 5 demonstrates experimental setup and results in which we analyze and compare overhead of estimation as well as its effectiveness on end-to-end packet delivery ratio.

2. RELATED WORK

A great deal of research has been conducted in the area of wireless link quality estimation and measurement. Here we touch three aspects of LQE schemes i.e. monitoring approach, measurement metric and implementation perspective.

Measurement approach for LQEs can be active or passive. Passive monitoring is performed on native network packets whereas active monitoring entails estimation over periodically sent control probe packets. Software based LQEs employ three main measurement metrics, namely, PRR, Required Number of Packet transmissions (RNP) and delay based schemes.

RNP based schemes ETX [5], HETX [4] and ETT [19] etc. send proactive broadcast beacon packets and measure the RNP from PRR of the periodic broadcasts to estimate link quality. For ETX based metrics, ETX (link) = $1 / (PRR_f \times PRR_r)$. PRR_f is the probability of successful (proactive broadcast) packet reception in the forward direction while the opposite direction is PRR_r. As mentioned earlier, broadcast packets have some major differences from unicast packets. These are reasons why estimations based on broadcasts have shown to perform poorly in varied and especially high traffic conditions [7]. Das et al. [25] performed comprehensive measurements over a two mesh test bed to study the instability in ETX link quality measurements. Their experiments showed that introducing a transfer of just one large file in the network resulted in link ETX value increasing up to 10000% suggesting much lower throughput than actually experienced by the link.

xDDR [7] is a PRR based metric that avoids the ETX broadcast problem by employing proactive unicast beacons to estimate link quality. xDDR is shown to be more accurate than broadcast based schemes. This however comes at the price of additional unicast control packet overhead.

Four-bit [12] is a hybrid estimator that uses both passive and active monitoring and is initiated at the sender. During active monitoring, nodes periodically broadcast probe packets used to compute approximation of the RNP.

Using delay as quality measurement metric is not a new phenomenon. Yet most delay based metrics have been designed to assist channel selection or timeliness requirements for real-time or multimedia based environments [21][22]. More recently, research related to relationship between delay and PDR is gaining attention [19].

The per-hop Round-Trip Time (RTT) [20] measures bi-directional delay (packet + ACK transmission time) on proactive unicast probe packets over a given link. RTT was designed to work with *Multi-radio Unification Protocol* (MUP) to assist high bandwidth channel selection. The protocol dictates the probes to be sent within high priority queue (available in IEEE 802.11e) in order to avoid queuing delay in the delay recording. While queuing delay is understandably irrelevant for channel selection, it plays a significant role in case of data driven routing.

Authors of [19] proposed a real-time End-to-End Delay Estimation Metric (EEDEM) that takes queuing delay into account as well. Delay accounting is accomplished by placing timers at parts of code in application, network, MAC and PHY layers from where the data packets pass through. The delays used are: 1) Generation Internal Delay (*GenIntDelay*) registered when packets are generated; 2) Forward Internal Delay (*FwdIntDelay*) registered when packets are being forwarded; 3) Receiving Internal Delay (*RecIntDelay*) registered when packets reach the destination. The *GenIntDelay* obtained at a node i with a parent p is calculated as follows:

$$GenIntDelay_{ip} = L5L3D_i + L3L2D_i + QueueD_i + TransD_{ip}$$

$L_gL_hD_i$ is the delay between layer g and layer h at the Node i, $QueueD_i$ is the MAC queuing delay, and $TransD_i$ is the transmission delay. The *FwdIntDelay* obtained at a node p with a parent s is calculated as:

$$FwdIntDelay_{ps} = FwdL2L3D_p + L3L2D_p + QueueD_p + TransD_{ps}$$

The *RecIntDelay* obtained at the sink s node is calculated as:

$$RecIntDelay_s = L2L3D_s + L3L5D_s$$

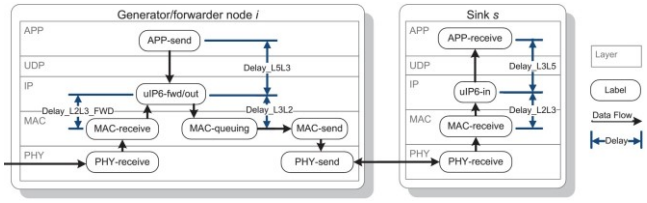

Figure 1. EEDEM Process Delay Calculation

EEDEM metric uses 9 timers placed at different layers to calculate queuing, transmission and processing delays. Figure 1 shows the processing delay calculation. Note that this approach requires modification in multiple layers which complicates the implementation process. More importantly though, EEDEM does not take into account back-off and retransmission delays. In contrast, we use 2 timers placed at network layer to account for 7 delays effecting communicating applications i.e. queuing delay, carrier sense delay, back-off delay, transmission delay, retransmission delay, processing delay and propagation delay. All these seven delays directly impact the data packets of communicating applications. For example, EEDEM nodes are not able to distinguish between two received packets from two different neighbors where neighbor one delivered the packet in first attempt while neighbor 2 delivered in the 4th attempt. Although, there is a possibility that the lossier link might have higher queuing delay but there is no substitute to more information especially when all these factors directly affect the data packets.

3. GENERIC NETWORK ATTRIBUTES

This section enlists preliminaries as well as the network architecture and attributes that are generic regardless to the quality optimization metric being employed. The network topology is modeled as a directed graph involving nodes and edges represented as (N, E) where $N = \{n_1, n_2, n_3, ...\}$ and $E = \{e_1, e_2, e_3, ... \}$. An edge represented as $<n_1, n_2>$ is a directional link from $n_1 \in N$ to $n_2 \in N$ ($<n_1, n_2> \neq <n_2, n_1>$) when node n_2 is within the transmission range of n_1 (1-hop neighbor). We use $\mathbb{N}$ to represent the set of natural numbers, $\mathbb{R}$ to represent the set of real numbers and $\mathbb{R}^{\geq 0}$ to represent the set of non-negative real numbers. Between a pair of in-range nodes such as n_1 and n_2, we compute the link-level quality estimate represented as $\lambda(n_1, n_2, t') \in \mathbb{R}^{\geq 0}$ with t' the current timestamp. Moreover, $\lambda(n_1, n_2, t')$ will often not be equal to $\lambda(n_2, n_1, t')$ due to the link asymmetry observed in wireless mobile communication [14]. For a given node n_k, we have $E_k(t') \subseteq E$ consisting of directional edges with the 1-hop neighbors of n_k at time t'. We have a set of connectionless UDP streams M, to transmit from a source to a specific destination node. A given stream $m \in M$ is a tuple:

$$m(n_s, n_d, f, t_{st}) \in N \times N \times \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$$

Figure 2 show the network topology at a given time instance t' where $t' > t'_3 > t'_2 > t_1$. Here, n_s is the stream source, n_d is the stream destination, f is the data rate of the transmission and t_{st} represents the start time. A stream transmission can only be initiated when there exist one or more possible routes from n_s to n_d. The set of available routes is represented as tuple:

$$m^{\tau}(n_s, n_d, f, t_{st}, t'_1) = \{\tau_1, \tau_2, \tau_3, \dots\}$$

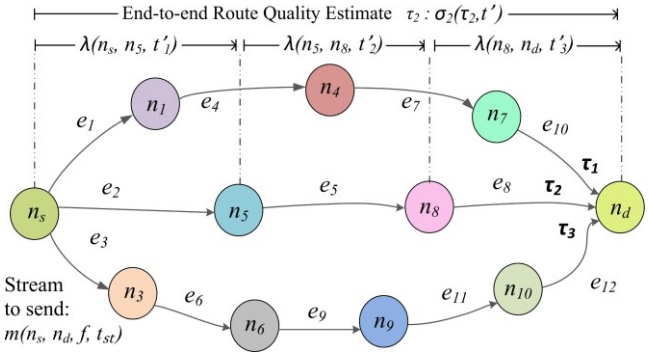

Figure 2. Network Topology

τ_1, τ_2 and τ_3 represent the distinct routes available to reach from the stream source n_s to destination n_d. Furthermore, $\sigma(\tau_2, t')$ is the end-to-end route quality estimate for a given route τ_2 at timestamp t', calculated from link-level quality estimates $\lambda(n_s, n_5, t'_1)$, $\lambda(n_5, n_8, t'_2)$ and $\lambda(n_8, n_d, t'_3)$ corresponding to all the edges present in the route τ_2. Depending on the quality metric in use, end-to-end quality estimate of a route may be the product, summation, minimum or maximum of the individual link qualities. Once a route is picked from the set of available routes $m^{\tau}(n_s, n_d, f, t_{st}, t'_1)$, n_s sends its first network layer packet $n_s \rightarrow n_5$ for stream destination n_d. Network communications are represented as follows:

$n_k \rightarrow n_l$: Unicast data packet/stream from node n_k to n_l

$n_k \Rightarrow *$: Broadcast control packet from n_k to neighbors

$n_k \Rightarrow n_l$: Broadcast control packet received at n_l from n_k

$n_s \rightarrow\!\!\!\rightarrow\!\!\!\rightarrow n_d$: Unicast data stream originating from stream source node n_s to stream destination n_d

A given network layer packet p is represented as follows:

$$p^{pid}(n_s, n_d, n_{pre}, n_{nxt}) \in N \times N \times N \times N \times N$$

pid represents the packet ID; n_{pre} and n_{nxt} represent the previous hop node and the next hop node respectively. For a given stream $m \in M$, $\mu(s, t_{t\Delta})$ denotes the total number of packets transmitted till timestamp t' where $t_{t\Delta} = (t' - t_{st})$. Similarly $\delta(m,t')$ represents the total number of packets successfully received by destination node n_d till time t' where $\delta \leq \mu$. End-to-end route quality is calculated at the destination node. For a given stream the end-to-end Packet Delivery Ratio (PDR) is:

$$PDR(m,t') = \frac{\delta(m,t_{t\Delta})}{\mu(m,t_{t\Delta})} \times 100\% \qquad (1)$$

In our network, the link-layer feedback is active and the sender's MAC layer attempts to retry packets that are unsuccessful in the first attempt for $(r-1)$ times where r represents the retry limit (including first attempt).

In summary, our goal is to equip communicating end-to-end nodes with a low cost link-level resource differentiation metric that improves their end-to-end multi-hop packet delivery.

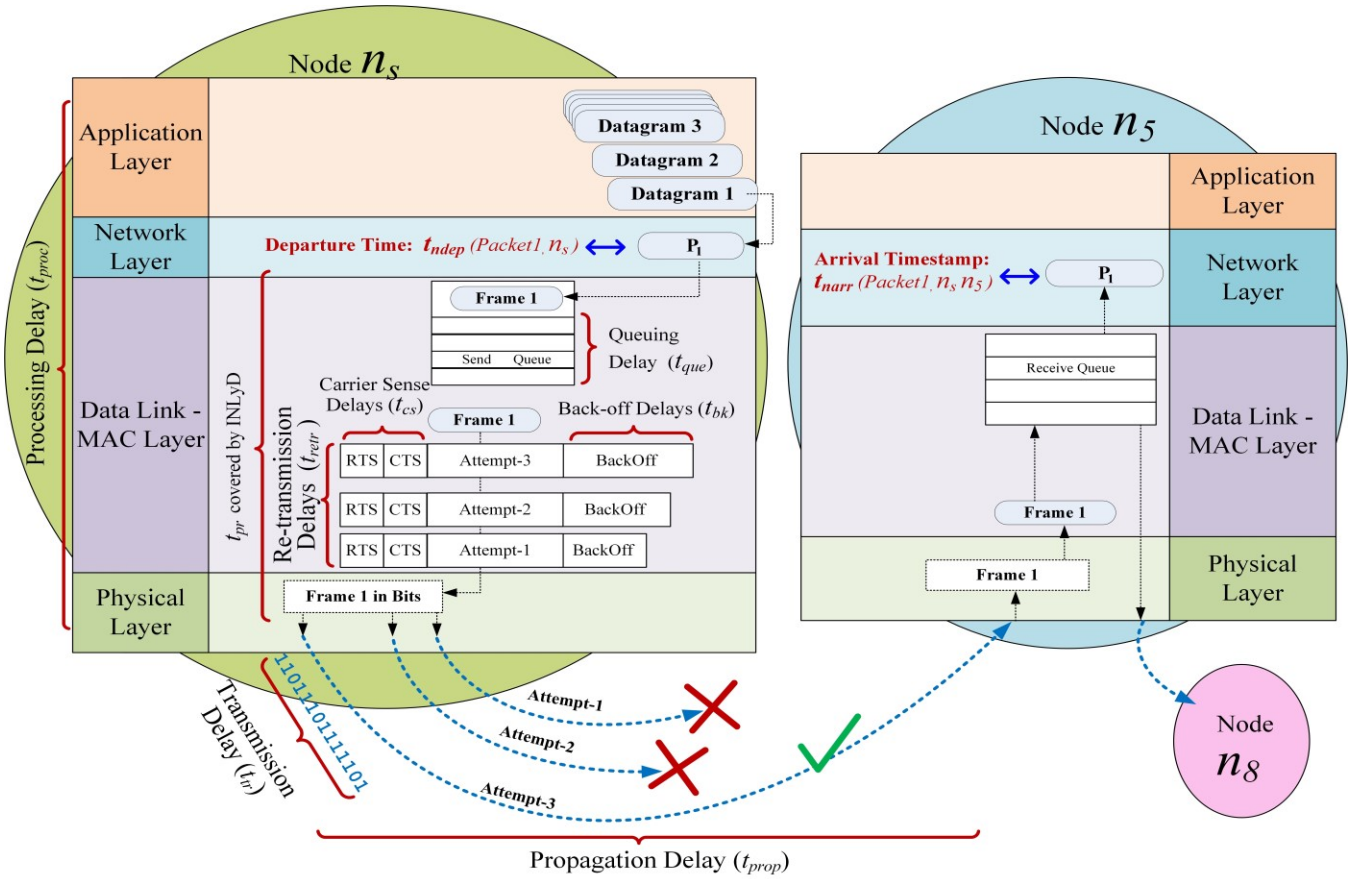

Figure 3. Inside inter-network-layer delay calculation

4. CUMULATIVE INTER-NETWORK-LAYER DELAY METRIC

This section details our proposed Inter-Network-Layer Delay (INLyD) metric as a low cost solution to finding high throughput multi-hop links among wireless devices. INLyD assumes that the communicating devices have clock synchronization which is a widely researched area [9]. INLyD comprises three main modules. The first entails Inter-Network-Layer delay calculation per packet i.e. in-band signaling of time-related information alongside data packets using the information at the downstream node. The second module is responsible for per link INLyD calculation. This entails managing freshness and integrity of INLyD entries. Prior to averaging the packet stats this module ensures freshness of per packet INLyD entries. The third is a quality-aware routing module that utilizes link-level estimates to attain a route-level estimate. In this regards, we describe INLyD metric's working with our extension of Dynamic Source Routing (DSR) protocol [16].

4.1 INLyD Calculation per Packet

Unlike active link monitoring schemes that estimate link statistics from proactively generated network or MAC layer control packets, INLyD monitors natively generated data and control packets by the nodes to estimate the respective link's quality. The majority of these packets are the natively generated data packets from applications but they also include natively generated control packets from the network layer and higher layers. First we look at the sender side (this implies stream source node or any intermediate forwarding node) where each packet p^{pid} originating from the particular node's network layer or above gets time-stamped with a departure time t_{ndep}. These include the TCP and UDP application data packets as well as control packets from network layer and higher layers. A given node n_k timestamps packets $\{p^1, p^2, p^3, \ldots\} \in P$ with departure times $t_{ndep}(p^1, n_k)$, $t_{ndep}(p^2, n_k)$ and $t_{ndep}(p^3, n_k)$. When the packet p^1 is received by a neighboring node n_l it is passed up to the network layer from the MAC layer. Node n_l attaches network-layer-arrival timestamp t_{narr} to the packet as $t_{narr}(p^1, n_k, n_l)$. This delay incurred by a given packet p^{pid} from the network layer of n_k to network layer of n_l thereby makes one individual entry of per packet inter-network-layer delay between the two neighboring nodes and is represented as $\underline{n}(p^{pid}, n_k, n_l)$. Likewise set $\underline{N}(n_k, n_l)$ represents the list of inter-network-layer delay entries $\underline{n}$ corresponding to all the packets received from n_k to n_l within a given time. Node n_l obtains $t_{ndep}(p^1, n_k)$ from the received packet and computes the INLyD entry for the first packet received as:

$$\underline{n}(p^1, n_k, n_l) = t_{narr}(p^1, n_k, n_l) - t_{ndep}(p^1, n_k) \quad (2)$$

Packet transfer between a pair of nodes experiences these seven delays namely processing delay, queuing delay, carrier sense delay, back-off delay, transmission delay, retransmission delay and propagation delay. With $\check{n}$ representing overall inter-nodal delay, we have:

$$\check{n} = t_{que} + t_{cs} + t_{bk} + t_{tr} + t_{retr} + t_{prop} + t_{proc} \quad (4)$$

Queuing delay (t_{que}) is the time a packet spends in the queue at the node waiting for its turn. From INLyD metric's perspective this includes both sender-side and receiver-side queue delays. Carrier

sense delay (t_{cs}) is incurred when a node with packets to send performs carrier sense e.g. Request-To-Send (RTS), Clear-To-Send (CTS) etc. Its value also depends on the contention window size. Back-off delay (t_{bk}) happens when either the carrier sense detects an ongoing transmission or a collision occurs with the RTS packet. Transmission delay (t_{tr}) is the time required to put the entire packet into the channel. It is determined by datarate, packet size and the coding scheme employed. In case the data packet's delivery fails it incurs retransmission delay (t_{retr}) to retry the packet again. Propagation delay (t_{prop}) implies the time it takes a signal change to propagate from physical layer of sender node to physical layer of the receiver. Processing delay (t_{proc}) refers to the computational time that occurs at the device level pertaining to handling and processing the data structures and related instructions. In wireless sensor devices, most time is consumed at the queuing, back-off and retransmissions; and most energy is consumed at sensing, transmission, retransmissions and propagation of signals/packets. Processing time in fact one of the least significant of the delays in terms of time spent. It is shown that energy consumption to transmit a single bit over the wireless channel is comparable to that consumed to carry out (depending on the platform) approximately 400 [15] to 800 [2] instructions.

Definition 1. As seen by the network layer, Inter-Network-Layer Delay (INLyD) is the delay a packet incurs from the time it enters sender side's queue till it leaves receiver side's queue.

Figure 3 shows how each instance of inter-network-delay quintessentially records six of the seven delays in (4) while also partially covering t_{proc} delay incurred by network layer and below. Application-level processing delays occur outside the INLyD metric's jurisdiction. However as mentioned above t_{proc} is much less significant compared to the other delays. We can see how the INLyD metric benefits from the processing hierarchy of network layer model. First we notice that the t_{ndep} is assigned by the network layer to p^1 i.e. t_{ndep} (p^1, n_s), therefore each data frame attempt (transmission or retransmission) of this packet includes the original departure time $t_{ndep}(p^1, n_s)$. When the packet is received by the node n_5 in its third attempt, the corresponding INLyD entry of $n_s \rightarrow n_5$ is:

$$\underline{n}(p^1, n_s, n_5) = t_{narr}(p^1, n_s, n_5) - t_{ndep}(p^1, n_s)$$

This in essence aggregates the processing delay partially and the rest of the delays entirely. Furthermore, a lossier link on average requires more retransmissions per successful network layer packet. This is in fact the basis for all ETX-based LQEs where a link's lossy-ness is estimated by the number of retransmissions the link requires for sending one successful packet. The lossier links therefore on average require more retransmissions per data packet; similarly, their overall average INLyD delay will be higher than healthier less lossy links. Therefore, such link will also have higher average inter-network-layer delay.

4.2 INLyD Averaging Module

INLyD averaging module is responsible for two main processes: *(1)* maintaining freshness of entries and *(2)* handling the shared channel and shared medium aspect of wireless links. INLyD computes inter-network-delay average at the downstream node over a variable length estimation window that keeps a chronological list of INLyD entries related to the particular link alongside each entry's corresponding network-layer time of arrivals t_{narr}. Since the communication data can also be sporadic in nature, INLyD's averaging window ($\hat{w}$) maintains freshness of entries with the combination of sliding and/or shrinking window feature. Which of the two actions is/are needed to be performed on the window is dictated by two configuration parameters i.e. INLyD window length ($\hat{w}_{len}$) and INLyD window duration ($\hat{w}_{dr}$). These parameters dictate two runtime attributes i.e. number of INLyD entries in window $|\hat{w}|$ and their corresponding list of network-layer arrival timestamps t_{narr} represented as (Ŧ). When | reaches allowed limit $\hat{w}_{len}$ When a new entry arrives for the same window (i.e. from the same upstream node) whose $|\hat{w}| = \hat{w}_{len}$, the INLyD window slides thereby discarding the oldest value from the front. $\hat{w}_{dr}$ dictates the freshness of the window in terms of time duration with help of timestamps t_{narr} of all packets received within the window. Using the combination of $\hat{w}_{len}$ and $\hat{w}_{dr}$ each given node automatically discards neighbor node entry if no fresh packets are received from it within $\hat{w}_{dr}$.

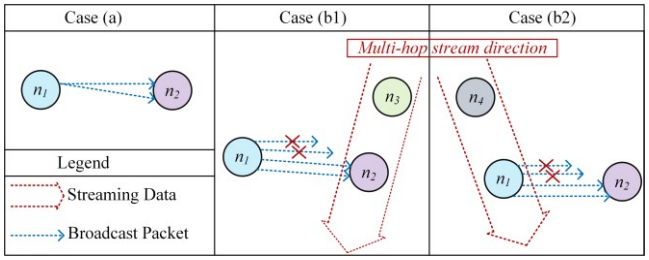

Figure 4. No-interference, shared medium and hidden terminal use cases

The fact that INLyD assesses a link's performance on the basis of natively generated communication alone raises the question as to how a link's quality is perceived when there is not enough historical native communication on the link. Little to no traffic received at network layer could mean one of two cases. *(Case a)* either there is simply too little traffic load over the given link as well as the node itself. *(Case b)* the link is experiencing excessive loss rate due to the shared nature of wireless medium resulting in collisions from neighboring traffic *(Case b1)* or hidden terminal [3] nodes *(Case b2)*. Making this distinction between the two cases is pivotal to INLyD's performance because case (a) in principle should suggest much less INLyD delay average as compared to the congested scenarios. Figure 4 illustrates the potential scenarios. All three cases show two successful broadcast packets freshly received from n_1 to n_2. From the network layer's view point, broadcast packets incur much less delay as compared to unicast packets primarily due to absence of retransmissions in broadcasted packets. Therefore unless case (a) can be correctly identified, all three cases will result in nearly equal delay average since the number of INLyD entries i.e. $|\hat{w}<n_1,n_2>| = 2$. This is why post-sliding/shrinking INLyD window whenever a node n_2 encounters $|\hat{w}<n_1,n_2>| \leq \hat{w}_{len}/\rho$; it checks whether there exist any link (e.g. $<n_3,n_2>$) with number of goodput entries within $\hat{w}_{dr}$ i.e. $|\hat{w}<n_3,n_2>| > \hat{w}_{len}/\rho$. $\hat{w}_{len}/\rho$ is the goodput lower bound and for our experiments we set it to ρ=10. Therefore the link-level quality estimate $\lambda(n_1, n_2, t')$ for case (a) and (b1) is as follows:

$$\lambda(n_1, n_2, t') = \begin{cases} \hat{w}_{avg}<n_1,n_2> & \text{case (a)} \\ Avg(\hat{w}_{avg}<n_1,n_2>, \hat{w}_{avg}<n_3,n_2>) & \text{case (b1)} \end{cases} \quad (5)$$

This way INLyD handles the shared channel and medium aspect at the receiver node shown as case b1 resulting in higher delay at $\hat{w}<n_1,n_2>$ for case (b1) compared to case (a). The congestion and collisions resulting from the sending node's neighborhood side i.e. case (b2) is more complicated and cannot be addressed at network layer without introducing some additional control handshake between n_1 and n_2. In principle however, channel sensing and RTS/CTS mechanism is present at the MAC layer to take care of

this problem. In other words, node n_1 will not send broadcast packets $n_1 \Rightarrow *$ if it senses the channel is busy with another streaming data or when n_1 itself sent CTS packet to n_4.

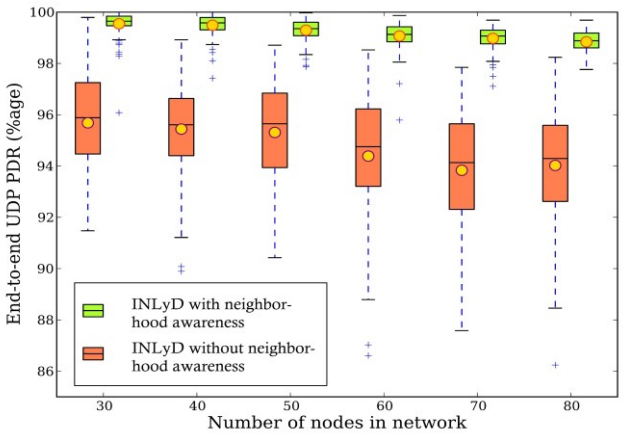

Figure 5. Comparison with and without nodal neighborhood awareness

Figure 5 shows the gain from accounting for the lossiness at the receiver side. We ran setups from 30 to 80 stationary nodes comprising 5 source-destination pairs transferring uniform UDP streams of CBR 30 i.e. 30 packets per second. The average end-to-end PDR yielded by INLyD with and without neighborhood awareness is calculated to observe the influence. For statistical correctness 20 trials were conducted for each network size each with different seeds resulting in a different network topology. The results were then averaged and compared. The yellow circular markers represent the average of a given network size i.e. average over 200 stream sender and receiver nodes (20 × (5 + 5)) per network size. We can see the benefit of integrating neighborhood quality awareness when determining a link's quality.

Algorithm 1 shows how the sliding and shrinking mechanism works alongside the averaging module computes the inter-network-layer delay over a given link $<n_k, n_l>$ at node n_k. We can also notice how INLyD by design over-estimates quality for in-range nodes that are not delegated a data stream yet.

Algorithm 1 INLyD Window Selection & Averaging

$\mathbf{E_k \subseteq E}$ → list of first hop edges of n_k at current time t'
$\underline{n}\langle p^{pid}, n_k, n_l\rangle$ → inter-network-delay of packet p^{pid} for $n_k \rightarrow n_l$
$\underline{N}\langle n_k, n_l\rangle$ → list of inter-network-layer delays $\underline{n}$ over the link $<n_k,n_l>$ (in reverse chronological order)
$\underline{\mathcal{T}}\langle n_k, n_l\rangle$ → list of corresponding packet arrival timestamps (in reverse chronological order)

```
while t'- t_narr(N<n_k,n_l>.front()) > ŵ_dr then
    N<n_k,n_l>.pop_front()
    T<n_k,n_l>.pop_front()
endwhile

ŵ<n_k,n_l> := N<n_k,n_l>
ŵ_avg<n_k,n_l> := SUM(ŵ<n_k,n_l>) / |ŵ<n_k,n_l>|

if |ŵ<n_k,n_l>| ≤ (ŵ_len/ρ) then
    count := 1
shared_inlyd := 0
for each e_i in E_k then
    ŵ<e_i> := perform_freshness(N<e_i>)
    if |ŵ<e_i>| > (ŵ_len/ρ) then
        shared_inlyd += SUM(ŵ<e_i>) / |ŵ<e_i>|
        count++
    endif
endfor
ŵ_avg<n_k,n_l> := (shared_inlyd + ŵ_avg<n_k,n_l>) / count
endif

return ŵ_avg<n_k,n_l>
```

4.3 INLyD Propagation & Route Selection

INLyD works together with a quality aware routing/gossiping module. For our research, we extend the Dynamic Source Routing protocol (DSR) [16] to work coupled with INLyD and other competing quality estimation metrics such as ETX, xDDR etc. However, as estimation metric INLyD is completely independent of the routing protocol, or the MAC or PHY layer for that matter. Any network layer routing protocol can be extended to attach t_{ndep} time to packets and the receiving node performs the averaging.

When a node n_s initiates route discovery to send a stream from source node n_s to n_d ($n_s \rightarrow\rightarrow\rightarrow n_d$), the cumulative INLyD - the sum of link delays - $\Sigma\lambda(\tau_{tid})$ is set to 0 at the start. Over the course of discovery, when an intermediate node n_l receives the discovery packet it looks up the previous hop node n_k and adds the respective link delay of its upstream neighbor $\lambda<n_k,n_l,t'>$ to $\Sigma\lambda(\tau_1)$ field in the discovery control packet. The discovery packet continues to propagate across the network until it reaches the intended destination of the stream n_d alongside the $\Sigma\lambda(\tau_1)$ of all the links traversed during the discovery by route option τ_1. Once the first discovery packet is received, n_d waits for time $t_{\tau\Delta}$ expecting to receive competing route options τ_1, τ_3 etc. If more route options reach the destination node n_d within $t_{\tau\Delta}$, it compares to see which route indicates the lowest cumulative INLyD delay and thereby selects the best route option τ_Θ for $n_s \rightarrow\rightarrow\rightarrow n_d$.

$$\forall \tau_u \in s^\tau(n_k, n_l, f, t_{st}),$$
$$\tau_\Theta = \tau_x \Leftrightarrow \Sigma\lambda(\tau_x) = Min_{\tau_u \in s^\tau}[\Sigma\lambda(\tau_u), (t_{\tau 1}, t_{\tau 1} + t_{\tau\Delta})] \quad (6)$$

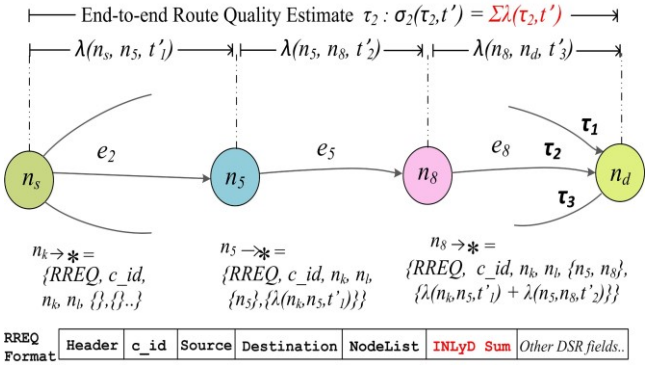

Figure 6. INLyD working with DSR Extension

Figure 6 shows our DSR extension and INLyD in route discovery (RREQ) operation. The three route options $\{\tau_1, \tau_2, \tau_3\}$ for transmission from n_s to n_d includes *nodeList* $\{n_1, n_4, n_7\}$, $\{n_5, n_8\}$ and $\{n_3, n_6, n_9, n_{10}\}$ respectively. A route τ can be decomposed

into *{source, nodeList, destination}*. In other words, the route τ_2 = $\{n_s, n_5, n_8, n_d\}$ where the expression $n_5 \in \tau_2$ indicates that node n_5 is included in the route τ_2. In our implementation setups node broadcasts route maintenance only in case of link breakage.

5. EXPERIMENTAL EVALUATION

To test and compare our quality estimation metric we employed OMNeT++ (Objective Modular Network Test-bed) simulator [11]. OMNET++ is an open-architecture, extensible, modular, component-based C++ discrete event simulation environment with strong GUI support and an embeddable simulation kernel. Due to its extensible nature, it is widely used for developing and testing large scale communication networks. OMNET++ offers a few extensions tailor-made for specific networking paradigms. For our experiments we used the INETMANET extension [12] which is specifically dedicated to MANETs and offers a variety of mobility models specifically related to MANET mobility. We extended the DSR implementation available in INETMANET extension to use our quality driven routing framework. We used INLyD, xDDR, ETX, HETX, Minimum Hop Count (MHC) as the link-level quality estimation metrics for our quality-aware routing module and compared their performance. Table 1 shows related network parameters generic to our setups.

TABLE 1. Network simulation parameters omnet++

Parameter	Value	Parameter	Value
Packet size	500 B	SimulationTime (t_{th})	200 s
Trials per experiment	20	Mobility Change Interval	100 s
Mobility Change Angle	normal (0deg, 90deg)	Mobility Speed	Uniform(0.5 mps,0.8mps)
INLyD Window length ($\hat{w}_{dr}$)	100	INLyD Window Duration ($\hat{w}_{dr}$)	10 s
Route Discovery wait time (t_{rt})	10 ms	ETX, HETX, xDDR send interval	0.1 s
Propagation Model	Path Loss Model	Radio Sensitivity	-90 dBm
MAC	802.11 g	MAC bitrate	54 Mbps
UDPStartTime	11 s	Radio Transmit Power	1.0 mW

We did not include EEDEM in our comparison since it attempts to do similar as INLyD but it does so accounting for lesser number of delay metrics and in a far less efficient manner. While it measures the processing, transmission and queuing delay, it is completely oblivious to the number of transmissions incurred for a given packet that it receives. Therefore, receiving nodes will not be able to distinguish between two senders that have similar processing, queuing and transmission delay but one with lot lower successful delivery rate than the other. For communicating applications, retransmission related delays are often far more significant than queuing, transmission and processing delays. Furthermore, authors propose placing 9 timers in multiple layers which is a very tedious and error prone approach. INLyD on the other hand measures every single delay that affects communicating nodes in a far simpler efficient manner.

To examine the role of a given LQE on impacting end-to-end delivery ratio we laid out a number of experimental setups. The setups comprised uniform traffic, varied traffic, varied network size, stationary as well as mobile scenarios. We compare the influence of employing different quality estimation metrics on delivery ratio of end-to-end UDP streams. In all our experiments we placed 5 stream source and 5 destination nodes at the cross-diagonal boundaries to each other on a 600m×600m terrain. The remaining intermediate nodes are randomly spread. Each source node is set to transmit a given stream destined for the destination node stationed on the opposite boundary. The average end-to-end PDR yielded by employing distinct LQEs is calculated to observe their influence. For statistical correctness 20 trials were conducted for each network size. Each trial used different seed resulting in a different network topology for the intermediate nodes. The results were then averaged and compared. The red circular markers represent the average of a given network size i.e. average UDP PDR recorded between 200 sender receiver nodes (20 × (5 + 5)) per network size. We compared between INLyD, ETX, HETX, xDDR and MHC as route end-to-end quality estimation metrics. HETX, ETX and xDDR beaconing interval is set to 0.1s. Underlying MAC retransmission limit is 6.

5.1 Setup 1: Static topology - uniform traffic

We first examine the behavior on static setup with simultaneous uniform streams. The goal of these experiments is to compare the effectiveness of the estimate on network with lesser spatio-temporal changes. All 5 stream sources initiate transmission request (RREQ) at 11s for a uniform transmission rate stream of 50 CBR sending 50 packets per second for all source destination pairs. The routes are refreshed every 30s by re-initiating RREQs. We compared the end-to-end PDR conceded by employing INLyD, ETX, HETX, xDDR and MHC as end-to-end quality metrics in Figure 7. On overall average xDDR yielded the highest PDR at 79%, followed by INLyD at 74%. HETX, ETX and MHC delivered 57%, 46% and 38% respectively. We notice that unlike other LQEs, INLyD exhibits the least drop in end-to-end delivery as network density increases. Network density affects active link quality estimators harshly since the number of proactive packet sender's increases. And since INLyD by design favors newer unused links, increase in network size only means additional route options.

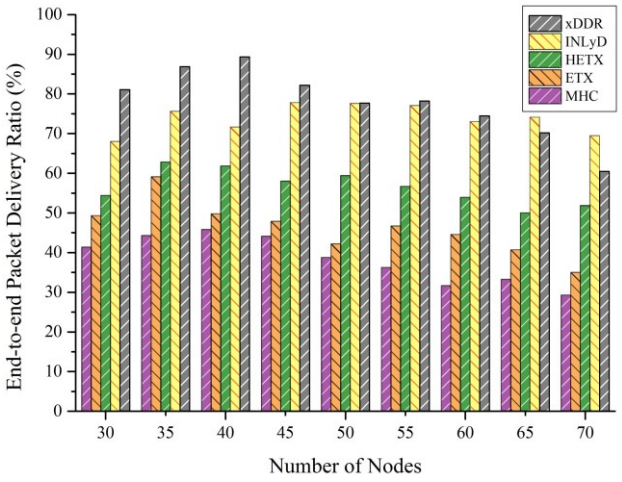

Figure 7. End-to-end UDP PDR in static environment without mobility (Setup 1)

We believe part of the reason why xDDR yielded higher PDR than INLyD is that in simultaneous start scenarios such as these by current time t' = 11s, INLyD metric has measured link performance on less than 10 broadcast packets such as RREQs. xDDR in comparison would assess link quality on approximately 110 unicast probe packets. This accurate estimation should mean that xDDR nodes are better at avoiding lossy links and therefore may experience lesser average collisions. However, in Figure 8

we see that xDDR metric's accurate estimation comes at the cost of generating more congestion and collisions in the network. Unlike ETX and HETX, xDDR transmits proactive packets as unicasts which results in additional collisions. INLyD on the other hand registers remarkably low collisions per node due to two main reasons. First, INLyD estimates link quality more accurately as compared to broadcast based schemes such as ETX and HETX. Second, it achieves this while generating no additional traffic in the network. In comparison, when a node in an active link monitoring environment is in range with 5 1-hop neighbors, ETX and HETX control packet from 1-hop neighbors at 0.1s will generate 50 packets per second on the shared channel. With xDDR the number is even higher depending on the retry limit and lossiness of the link. The impact of these extra control packets becomes more evident as the network size grows. Note that with HETX and xDDR collision rate rises sharply in comparison as the node density increase. Overall INLyD nodes experience 37% lesser collisions compared to that of HETX and xDDR nodes.

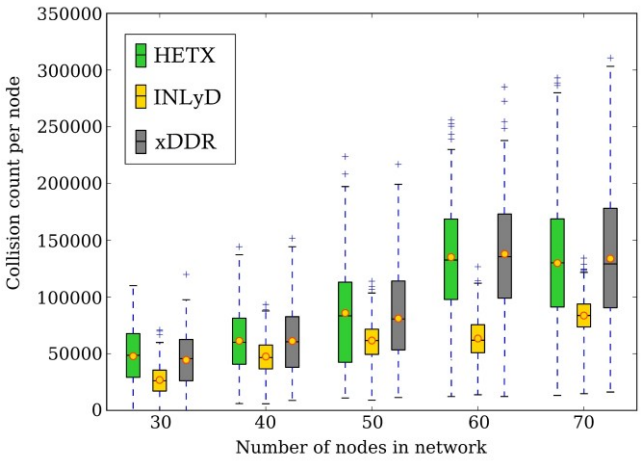

Figure 8. Average number of collisions recorded per node per 20 trials (Setup 1)

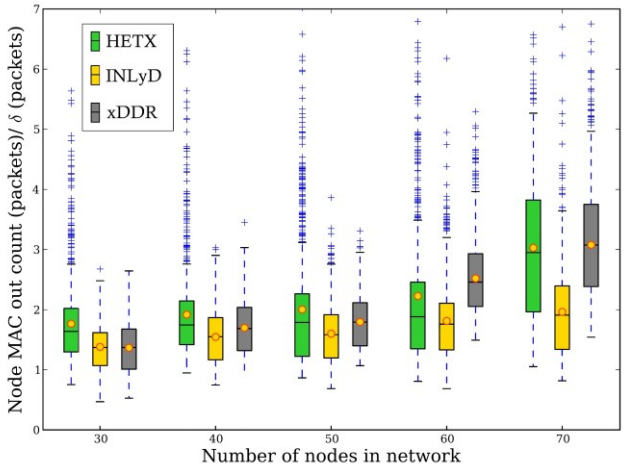

Figure 9. Goodput comparison. Ratio of total no. of packets transmitted by MAC layer and total UDP packets delivered end-to-end (Setup 1)

In MANETs and WSNs, packet transmission is one of the most energy intensive processes. In Figure 9 therefore we compare the goodput of each of these schemes. Given a quality metric, we measure the ratio of total number of MAC transmissions (MAC out count) per node (including collisions, retransmissions etc.) w.r.t. total number of UDP packets delivered end-to-end. As the network density increases, so does the congestion. At network size 60 and 70, HETX underperforms due to wavered link quality estimates as the network density grows. Picking a lossy route inaccurately as the best route results in data being transmitted through links that may require more retransmissions per successful packet delivery than other rightful options. In case of xDDR we believe this increase is the effect of increased number of MAC out packets resulting from proactive unicast probe retransmissions and their respective collisions with data packets contributing to the increased MAC out count. INLyD on the other hand shows less the least impact of increased node density. In its best case (70 nodes), nodes in network running INLyD expend 35% and 33% less number of MAC transmissions per successful end-to-end UDP delivery. On average (network size 30 to 70), INLyD network nodes spent 22% and 27% less MAC packets than HETX and xDDR respectively.

5.2 Setup 2: Mobile topology - uniform traffic

In mobility scenarios, we aimed to emulate movement of people with smartphones in outdoor terrains such as disaster relief camps and festivals etc. For this purpose, we reviewed a number of mobility models presented in [13]. These include Random Waypoint mobility, Gauss-Markov mobility, Mass Mobility (MM) and Chiang mobility models. We found MM to be best fit for the scenario in terms of mobility and movement characteristics. In MM, mobile nodes move within area specified in the terrain. Nodes move along a straight line for a specific duration and then make a turn. This duration of movement in straight line is controlled by parameter changeInterval which takes duration and standard deviation (changeInterval = 100 s, Standard Deviation SD = 1 s in our setup). The turn angle to dictate the new direction after every changeInterval is a normally distributed random number with average equal to preceding direction and standard deviation (SD = 90 degrees in our setup). Similarly, after each changeInterval, the node speed is taken as a uniform distribution within a range of speeds. Our setup used 1.8 km/h to 2.9 km/h.

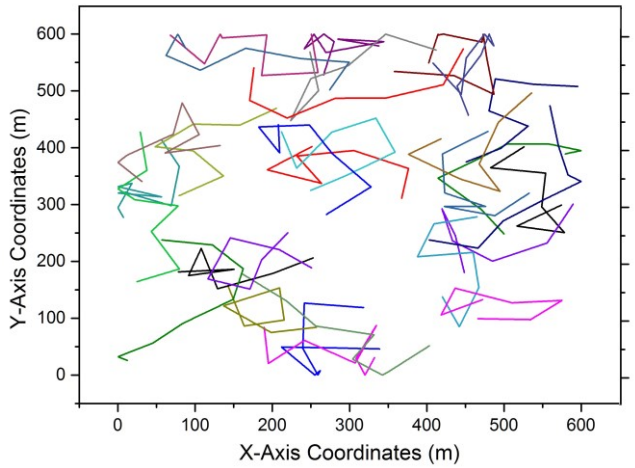

Figure 10. Mass mobility movement trajectory of 30 of nodes in a 600m×600m terrain during a run of 600s selecting new angle and speed every 100 seconds (each line represents an individual node) (Setup 2)

Figure 10 shows the aerial view of the movement pattern of 30 nodes in a 600m×600m terrain where line represents an individual node's movement pattern during 600s. Since the changeInterval is

set to 100s, we can see the 6 instances where change in angle of movement.

We ran the mobility scenario with network size of 50 comprising 5 source-destination pairs communicating uniform UDP streams at the rate of CBR 50. Results are averaged for 20 experimental trials where each trial produces different topology for intermediate nodes. The stream source and destination nodes are set as fixed while the intermediate nodes move in Mass Mobility movement pattern. Figure 11 shows the end-to-end PDR. Overall mobility setup with INLyD yields 72% end-to-end UDP stream PDR as opposed to 75% in the case of xDDR. ETX, HETX and MHC resulted in 60%, 57% and 50% respectively.

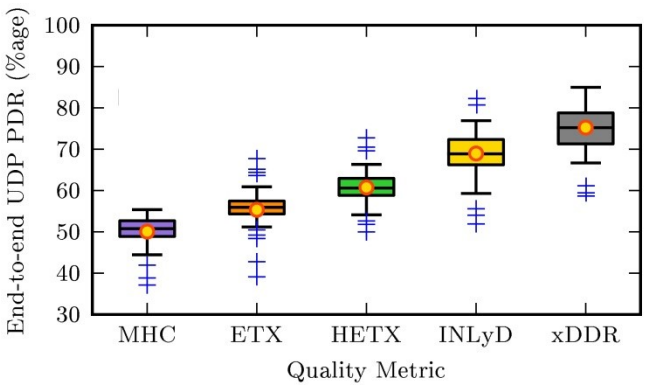

Figure 11. End-to-end UDP Data Delivery Ratio in mobility scenario with Mass mobility (Setup 2)

5.3 Setup 3: Static topology - Varied traffic load

Figure 12 shows the end-to-end PDR comparison for the varied traffic case on a network comprising 50 nodes. In this setup all 5 stream senders have varied stream start time (t_{st} = 11s, 21s, 31s, 41s, 51s) as well as varied data rate (f = 20, 30, 40, 50, 60). In varied traffic environment INLyD on average outperforms xDDR with average 82.5% end-to-end PDR as opposed to 79% in xDDR setup. Furthermore, ETX, HETX and MHC resulted in 66%, 60% and 43% end-to-end packet delivery respectively.

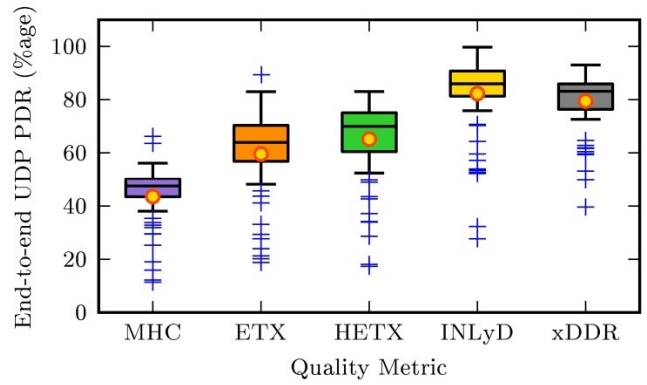

Figure 12. End-to-end UDP Data Delivery Ratio in variable data-rate traffic load and varied start times case with no mobility (Setup 3)

6. CONCLUSION

In this paper, we aim at devising a low cost, passive monitoring link quality metric that is independent of the underlying MAC layer. INLyD is a network layer metric that incorporates queuing, sensing, contention, transmission, retransmission, processing and propagation delay to evaluate link quality. More importantly though, INLyD relies solely on natively generated data and control packets to assess quality, and does not generate any additional control packets. We extended DSR to work with various quality driven metrics. Our results show INLyD to outperform contemporary QoS metrics yielding higher end-to-end packet delivery in static, mobile, uniform as well as varied traffic scenarios. With accurate estimations and no control packet overhead INLyD results in lower collisions and total number of transmissions per node in comparison. This consequently entails lower packet transmissions needed per data packet delivery thereby significantly improving device lifetime. These characteristics make the INLyD metric an ideal candidate for network environments such as disaster relief MANETs or long term deployed WSNs.

7. ACKNOWLEDGMENTS

This work was supported by the SenSafety project in the Dutch Commit program, www.sensafety.nl.

8. REFERENCES

[1] R.J. Ellison, D.A. Fisher, and R.C. Linger, "Survivable Network System: An Emerging Discipline". Technical Report, CMU/SEI-97-TR-013, Carnegie Mellon University, 1997.

[2] M. Samuel, F. Michael, H. Joseph, W. Hong. "TAG: A tiny aggregation service for ad-hoc sensor networks". In OSDI. Citeseer, 2002.

[3] A. Jayasuriya, S. Perreau, A. Dadej, and S. Gordon. "Hidden vs exposed terminal problem in ad hoc networks." PhD dissertation, ATNAC 2004, 2004.

[4] A. T. Tran, and M. K. Kim, "Characteristics of ETX Link Quality Estimator Under High Traffic Load in Wireless Networks", 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing HPCC_EUC 2013, pp.611,618, 13-15 Nov. 2013

[5] D. Couto, D. Aguayo, J. Bicket and R. Morris, "A high-throughput path metric for multi-hop wireless routing", Wireless Networks, vol. 11, no. 4, pp. 419–434, 2005.

[6] X. Ni, Performance evaluation of ETX on grid based wireless mesh networks, MPhil Thesis Report EET-UNSW. University of New South Wales, Australia, 2008

[7] S.R. Afzal, M. Nabi, S. Stuijk and T. Basten, "Improving end-to-end packet delivery in high traffic multi-hop wireless ad hoc networks". In Proceedings of the 8th International Conference on Mobile Multimedia Communications, pp. 39-46, 2015.

[8] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four bit wireless link estimation," In Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets VI), 2007.

[9] Y.-C. Wu, Q. Chauhari and E. Serpedin, "Clock synchronization of wireless sensor networks," IEEE Signal Processing Magazine, vol. 28, pp. 124-138, 2011.

[10] G. Prakash, and E. Anand. "Indian News Media and Natural Calamities: Case of Chennai Floods." International Journal of Multidisciplinary Approach & Studies 3.2 (2016).

[11] OMNeT++ Network Simulation Framework, http://www.omnetpp.org/ [accessed June 2016].

[12] INETMANET Extension for OMNET++, https://github.com/inetmanet/inetmanet [accessed June 2016].

[13] T. Camp, J. Boleng, and V. Davies. “A survey of mobility models for ad hoc network research.” Wireless communications and mobile computing 2.5 (2002): 483-502.

[14] J. Zhao and R. Govindan. “Understanding packet delivery performance in dense wireless sensor networks.” In the Proceedings First ACM Sensys Conference, November 2003.

[15] M. G. C. Torres, “Energy Consumption In Wireless Sensor Networks using GSP,” Dissertation, University of Pittsburgh, 2006.

[16] D. Johnson and D. Maltz, “Dynamic source routing in ad-hoc wireless networks.” Proc. of SIGCOMM '96, Aug. 1996.

[17] N. Baccour, A. Koubaa., L. Mottola, M.A. Zuniga, H. Youssef, C.A. Boano and M. Alves, 2012. “Radio link quality estimation in wireless sensor networks: a survey.” ACM Transactions on Sensor Networks (TOSN), 8(4), p.34.

[18] R. Draves, J.Padhye, and B.Zill, “Routing in multi-radio, multi-hop wireless mesh networks”, in the proceedings of MobiCom 2004, pp.114-128.

[19] P. Pinto, A. Pinto, M. Ricardo, “End-to-end delay estimation using RPL metrics in WSN.” In: IFIP Wireless Days (WD), pp. 1–6 (2013).

[20] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. “A multi-radio unification protocol for IEEE 802.11 wireless networks.” In Proceedings of Broadnets ’04, pp. 344 – 354, 2004.

[21] K. S. Kim, C. Li, and E. Mondiano, “Scheduling multicast traffic with deadlines in wireless networks,” Proceedings of IEEE INFOCOM, 2014.

[22] P. Jayachandran and M. Andrews, “Minimizing end-to-end delay in wireless networks using a coordinated edf schedule,” in Proc. of IEEE INFOCOM, 2010.

[23] K. H. Kim and K. G. Shin, 2006. “On accurate measurement of link quality in multi-hop wireless mesh networks.” In Proc. of the 12th Annual Int. Conf. on Mobile Computing and Networking (MobiCom ’06). ACM, 38–49.

[24] D. Lal, A. Manjeshwar and F. Herrmann, 2003. “Measurement and characterization of link quality metrics in energy constrained wireless sensor networks.” In Proc. of the IEEE Global Telecommunications Conference (Globecom ’03). IEEE Communications Society, 446–452.

[25] S. M. Das, H. Pucha, K. Papagiannaki, and Y. C. Hu, “Studying wireless routing link metric dynamics,” in Proceedings of ACM SIGCOMMIMC ’07, October 2007, pp .327–332.