LEDBAT Performance in Sub-packet Regimes

Ioannis Komnios
Space Internetworking Center
Dept. of ECE, Democritus University of Thrace
Xanthi, 67100, Greece
i.komnios@ee.duth.gr

Arjuna Sathiaselvan and Jon Crowcroft
Computer Laboratory
University of Cambridge
Cambridge, United Kingdom
{arjuna.sathiaselvan, jon.crowcroft}@cl.cam.ac.uk

Abstract—Wireless Community Networks in the developing world satisfy the basic needs of remote users to information access. However, community networks in developing regions usually rely on low-bandwidth backhaul links that are shared amongst a large user base, driving these links to sub-packet regimes where the per-flow throughput is less than one packet per RTT. TCP performance significantly degrades in such conditions, resulting in severe unfairness and high packet loss rates. In this paper, we investigate the performance of scavenger transport methods, namely LEDBAT and its fair modification fLEDBAT, in the sub-packet regime of shared backhaul links in developing regions. Our intention is to explore the feasibility of using such scavenger transport methods for uploading content over bandwidth constrained backhauls. Our findings show that LEDBAT achieves higher link efficiency and fairness compared to TCP in a variety of sub-packet regime scenarios. When TCP and LEDBAT flows share the same link in the sub-packet regime, LEDBAT flows are more aggressive, consuming more resources than TCP. Therefore, we conclude that a more conservative strategy after consecutive timeouts and shared bottleneck detection mechanisms need to be incorporated into the core LEDBAT algorithm, in order to correctly adjust its congestion window in the sub-packet regime.

Keywords—TCP; less-than-best-effort service; scavenger transport method; LEDBAT; congestion control; low bandwidth networks; sub-packet regime; developing regions

I. INTRODUCTION

The growth in the popularity of the Internet during the last decade is unprecedented. Internet is now part of everyday life in all parts of the developed world. However, providing sustainable, cost-effective and high-quality Internet connection with coverage for all citizens is still a challenging problem. Access problems often result from sparsely spread populations living in physically remote locations; it is simply not cost-effective for Internet Service Providers (ISPs) to install the required infrastructure for broadband Internet access to these areas. Coupled with physical limitations of terrestrial infrastructures to provide last mile access, remote communities also incur higher costs for connection between the exchange and backbone network mainly due to distance.

Wireless Community Networks (WCNs) have attracted the attention of stakeholders and non-profit organisations as a solution to provide cost-effective Internet access in emerging regions. Given the physical and economic constraints, typical users of wireless community networks share a low-bandwidth backhaul Internet link. With the ubiquitous availability of low-cost mobile devices, cloud-storage and applications that capitalize on the benefits offered by the cloud, and the increase in the complexity of websites, backhaul links in WCNs are often driven into the sub-packet regime, an environment where the per-flow throughput is less than one packet per round-trip time (RTT). The sub-packet regime has only recently attracted the research community, after findings showed that the performance of TCP seriously degrades in the sub-packet regime [1].

Scavenger or Less-than-Best-Effort (LBE) access methods based on the Lowest-Cost Denominator Network (LCD-Net) [2] paradigm have been recently proposed, in an effort to share the unused capacity of backhaul links with the underprivileged, without affecting the performance of paid customers. Such methods are typically less aggressive than TCP, consuming significantly less resources. One of the most popular Scavenger transport, namely LEDBAT, has been developed by BitTorrent, Inc. and is already implemented in the BitTorrent application. The performance of LEDBAT and its interaction with TCP flows in well-connected environments has been investigated previously and a few shortcomings have been identified [3]-[13].

In this paper, we present a performance analysis of LEDBAT in the sub-packet regime of shared backhaul links of wireless community networks in emerging regions. Our findings show that LEDBAT achieves higher link efficiency and fairness when compared to TCP in a variety of sub-packet regime scenarios. When TCP and LEDBAT flows share the same link in the sub-packet regime, LEDBAT flows fail to measure the actual base delay due to the standing queue and become aggressive, consuming more resources than TCP. We also confirm that active queue management (AQM) is not a solution to sub-packet regime; in contrast, AQM cancels that gains of LEDBAT by making them more aggressive. Shared bottleneck detection mechanisms need to be incorporated into LEDBAT, in order to correctly adjust its congestion window in the sub-packet regime. Moreover, LEDBAT needs to react more conservatively after consecutive timeouts, which are typical in the sub-packet regime.

The structure of the paper is organised as follows: in Section II we discuss the required background on wireless community networks in developing countries, the sub-packet regime, and typical scavenger transport methods. Section III describes our experimental methodology, while in Section IV we present our simulation results. A discussion and
recommendations section follows in Section V. We conclude
the paper in Section VI.

II. BACKGROUND

A. Wireless Community Networks in Developing Countries

The socio-economic development of rural regions in the
third world highly depends on access to information. Wireless
Community Networks have been proposed to connect rural
communities to the Internet. These networks are usually run by
non-profit organizations and can cooperate with local
stakeholders to develop community services, including local
networking, voice connections and Internet access. Rural
wireless networks usually share a low-bandwidth (with high
level of loss rates and packet reordering [14][15]) costly link to
the Internet amongst a large user base. For example, there
exists a large number of low-bandwidth community network
environments in the developing world with high levels of
network sharing where a 128Kbps–2Mbps access link may be
shared by 50–200 users [1].

One of the largest rural wireless mesh networks is the
AirJahdi network [16] in the Himalayan mountains of Northern
India, which connects 10,000 rural users with several long
distance links connecting schools, hospitals and offices. In
South Africa, the Peebles Valley mesh network [17] consists of
long distance wireless links covering 15 square kilometers and
the Internet bandwidth is provided by the an HIV/AIDS clinic
which has a sponsored very small aperture terminal (VSAT)
providing 256kbps/64kbps connection since no other
connectivity options were available. Similarly, the LinkNet
mesh network [18] is located in the very remote village of
Macha, Zambia. A distance of 70km from a tarred road or
landline phone, Macha provides rural connectivity to the
John’s Hopkins Malaria Institute at Macha, the Macha Mission
Hospital and the community at large within Macha.

B. Sub-packet Regime

Existing congestion control schemes, such as TCP-
NewReno, assume the fair-share bandwidth of a flow is at least
1 packet per RTT [19]. However, there exists a large number of
low-bandwidth community network environments in the
developing world with high levels of network sharing [20][21].
Sub-packet regime is defined as an environment where the
throughput is less than one packet per RTT. A flow with
segment size S and round-trip time of RTT is in the sub-packet
regime if both of the following conditions hold at the
bottleneck link, which has capacity C:

1. Number of competing flows, N \geq 1, and
2. Per-flow fair share is less than S/RTT.

The sub-packet regime is the result of heavy sharing, on the
order of several competing flows operating over low-
bandwidth networks. TCP and other common congestion
control protocols break down in the sub-packet regime,
resulting in severe unfairness, high packet loss rates, and
repetitive timeouts [22]-[24]. In addition, none of the standard
TCP variants or known queuing mechanisms offer substantial
performance gains in the sub-packet regime. The sub-packet
regime has not been a traditionally important region of
operation for network flows, and as a result this space has
remained relatively unexplored. The concept of a sub-packet
regime arises in prior work in the context of understanding the
behavior of TCP in the face of many competing flows [25].

The authors of [25] propose an analytical model to
characterize the equilibrium behavior of TCP in the sub-packet
regime. The model is a simpler variant of a full Markov model
for TCP operating in traditional regimes [26], but gives more
careful attention to modeling repetitive timeouts, an extremely
common state experienced by TCP flows in sub-packet
regimes. The proposed model can be used by network middle-
boxes in practice to enhance TCP performance and fairness in
sub-packet regimes.

C. Scavenger/Less-than-best-effort Transport Methods

TCP is the transport protocol of choice for most Internet
applications today. One of its main characteristics is that it
treats all flows equally in a best-effort manner, achieving flow-
rate fairness. According to [27], global mobile data traffic
will continue its truly remarkable growth, increasing 13-fold
over the next five years, while average global mobile network
speeds will increase seven-fold from 2012 (0.5 Mbps) to 2017
(3.9 Mbps). This calls for better resource utilization and
distribution among flows [28][29]. Scavenger transport
methods have been proposed as a solution to this problem. In
essence, non-real-time flows that can withstand a certain
amount of delay can delay their transmission, by being less
aggressive and consuming fewer resources than their fair-share,
leaving more capacity to real-time flows. Authors of [30]
provide a survey of transport protocols and congestion control
mechanisms that are designed to have a smaller bandwidth
and/or delay impact on standard TCP than standard TCP itself.

One of the most popular scavenger transport methods,
namely LEDBAT, has been proposed by BitTorrent, Inc., and
is already an experimental RFC [31] by the Internet
Engineering Task Force (IETF). LEDBAT has been designed
for use by background bulk-data applications, such as peer-to-
peer file transfers, to limit congestion that each flow itself
induces in the network. The congestion algorithm of LEDBAT
is illustrated in (1), (2) and (3). First, the current queuing delay
(\text{queuing delay}) is estimated by subtracting the minimum
delay (\text{base delay}) from the measured one-way delay
(\text{current delay}) in (1). Then, distance \(\Delta(t)\) from a predefined
target queuing delay (\text{TARGET}) is calculated in (2). If there is
no packet loss, the congestion window is recomputed based on
the upper part of (3), where \(\alpha\) is the increase/decrease factor. If
there is a packet loss, LEDBAT performs like TCP by halving
the congestion window.

\[
\text{queuing delay} = \text{current delay} - \text{base delay} \quad (1)
\]

\[
\Delta(t) = \text{queuing delay} - \text{TARGET} \quad (2)
\]

\[
cwnd(t+1)= \begin{cases} 
\frac{cwnd(t)+\alpha\times\text{TARGET} - \Delta(t)}{2}, & \text{if no loss} \\
\frac{cwnd(t)}{2}, & \text{if loss}
\end{cases}
\quad (3)
\]
The performance of LEDBAT, the tuning of its parameters, its interaction with TCP flows and its comparison to other popular scavenger transport have been investigated [3][4][7][9][10]. Extensive evaluations showed that LEDBAT presents a few malfunctions [6][8][12], with the most important being the so-called “late-comer advantage”, where a second, newly starting LEDBAT flow can starve the first, already running one. Several solutions have been proposed to solve this problem, with the most prominent being fLEDBAT, a modification to the LEDBAT algorithm that introduces multiplicative decrease of the congestion window continuously driven by the estimated distance from target [8][11][13]. The congestion algorithm of fLEDBAT is illustrated in (4). If Δ(t) is negative or zero, we have not yet surpassed the target delay, thus there is room for increase and the congestion window is additively increased (α is the increase factor). If Δ(t) is positive, it means that measured one-way delay is already larger than target delay, thus we need to multiplicatively reduce the congestion window (ζ is the decrease factor). If there is a packet loss, fLEDBAT performs like TCP by halving the congestion window.

\[
cwnd(t+1) = \begin{cases} 
\frac{cwnd(t)+\alpha}{cwnd(t)} & \text{if } \Delta(t) < 0 \\
\frac{cwnd(t)}{\frac{1}{\Delta}TARGET} & \text{if } \Delta(t) > 0 \\
\frac{1}{2} \cdot cwnd(t) & \text{if loss}
\end{cases}
\] (4)

LEDBAT performance has been investigated in a variety of simulation scenarios in large bandwidth delay product scenarios [32], as well as real implementations [5]. All work on LEDBAT so far has been focused on scenarios where the network is assumed to have sufficiently large capacity and is never driven into the sub-packet regime. To the best of our knowledge, this is the first time LEDBAT and fLEDBAT are evaluated in the sub-packet regime.

### III. METHODOLOGY

The aim of our simulations is twofold: first, to evaluate the performance of TCP NewReno, LEDBAT and fLEDBAT in the sub-packet regime, and, second, to investigate the unfairness issues that may arise when both TCP and fLEDBAT flows share the same link in the sub-packet regime.

#### A. Simulation Scenarios

The performance of TCP NewReno, LEDBAT and fLEDBAT is evaluated through simulations using ns-2 version 2.35 [33]. LEDBAT code was available [34], while fLEDBAT was implemented by modifying LEDBAT code. As reference network scenarios, we consider three different backhaul links with varying capacity (C) and delay, as depicted in Table I. The characteristics of the backhaul links have been extracted from measurements reported in [35][36]. The backhaul uplink is occupied by an increasing number of flows (N) ranging from 2 to 96, which is realistic considering typical backhaul links in emerging regions [1]. In all cases, buffer size is set equal to the bandwidth-delay product and all flows use a fixed packet size equal to 1500 Bytes, including 40 Bytes header. Parameter α is set to 1, while parameter ζ of fLEDBAT is set to 5. Both DropTail and RED without ECN [37] are used for queue management; when RED is used, maximum threshold is set equal to buffer size divided by two, while minimum threshold is set equal to maximum threshold divided by three. All flows are FTP, we randomize the start times of each flow uniformly between 0s and 30s and each simulation lasts for 300 seconds. Given the bandwidth-delay product of each scenario, the number of parallel flows might not be enough to drive the link into the sub-packet regime. For this reason, each figure is separated into two parts through a vertical dotted line: in the left part the network has not reached the sub-packet regime, while in the right part the number of flows is enough to drive the network in the sub-packet regime. Each set of experiments is repeated 30 times and the mean values for each evaluation metric are extracted. All simulations were also repeated using short FTP flows that send different file sizes instead of long-lived FTP flows.

#### B. Evaluation Metrics

We evaluate performance using a variety of metrics, such as link efficiency, fairness index, packet loss probability and queuing delay index. Each of the aforementioned metrics is defined as follows:

1. **Average link efficiency (η)** expresses the average link utilization as the ratio between the throughput sum of all flows, over the available capacity.

\[
\eta = \frac{\sum_{i=1}^{C} \text{Throughput}(i)}{C}
\] (5)

2. **Average Jain’s fairness index (F)** determines whether flows are receiving a fair share of the available resources.

\[
F = \frac{\left(\sum_{i=1}^{C} \text{Throughput}(i)^2\right)^{1/2}}{N \sum_{i=1}^{C} \text{Throughput}(i)}
\] (6)

3. **Average packet loss probability (P_l)** is calculated as the average ratio of dropped packets over the total number of packets sent over the link.

\[
P_l = \frac{\text{Total packets dropped}}{\text{Total packets sent}}
\] (7)

4. **Average queuing delay index (D_q)** is computed normalizing the mean queuing delay during the simulation over the maximum theoretical queuing delay.

\[
D_q = \frac{\text{Average Queuing Delay}}{\text{Maximum Queuing Delay}}
\] (8)

5. **Average traffic load distribution** is also calculated when flows that utilize different access methods coexist in the same link.

### Table I. BACKHAUL LINK CHARACTERISTICS FOR EACH SCENARIO

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Uplink Capacity</th>
<th>Downlink Capacity</th>
<th>RTT</th>
<th>Uplink Buffer</th>
<th>Downlink Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>58 Kbps</td>
<td>135 Kbps</td>
<td>900 ms</td>
<td>4 pkts</td>
<td>10 pkts</td>
</tr>
<tr>
<td>(b)</td>
<td>600 Kbps</td>
<td>1.2 Mbps</td>
<td>450 ms</td>
<td>22 pkts</td>
<td>45 pkts</td>
</tr>
<tr>
<td>(c)</td>
<td>800 Kbps</td>
<td>2.2 Mbps</td>
<td>450 ms</td>
<td>30 pkts</td>
<td>82 pkts</td>
</tr>
</tbody>
</table>
IV. RESULTS

Simulations are divided into two sets: (i) all flows use either TCP NewReno, LEDBAT or fLEDBAT as an access method; (ii) flows are equally divided into TCP and fLEDBAT flows, in order to investigate the operation of fLEDBAT in the presence of TCP flows in the sub-packet regime.

A. Comparison of TCP, LEDBAT and fLEDBAT for increasing number of flows

We start our analysis by considering an increasing number of flows that use the same access method (i.e. TCP NewReno, LEDBAT and fLEDBAT) and we repeat the experiments using both DropTail and RED as queue management algorithms, in order to validate the impact of active queue management on low priority congestion control as briefly described in [38], as well as different $\zeta$ values.

1) Average link efficiency

First, in order to demonstrate the effect of increasing traffic load on link utilization, we plot link efficiency for the uplink. Fig. 1 shows that the uplink is almost always fully utilized in all scenarios. When we are not in the sub-packet regime, fLEDBAT underutilizes bandwidth that can be exploited by other flows. When we enter the sub-packet regime, we notice that both LEDBAT and fLEDBAT achieve higher link efficiency than TCP. In essence, TCP tries to transmit a significant amount of data, but fails due to the fully utilized link, resulting in timeouts and retransmissions. LEDBAT and fLEDBAT do not timeout as often as TCP by being less aggressive and transmitting less data. This leads to significantly less retransmitted packets and higher link efficiency. The results are similar when we use RED as an active queue management algorithm and are thus omitted.

2) Average Jain’s fairness index

To explore the effect of increasing traffic load on the fairness of each protocol, we plot the average Jain’s fairness index of 30 runs for an increasing number of flows for all three backhaul links. Fig. 2 shows that in all cases fairness among fLEDBAT flows is significantly higher than TCP NewReno flows; by being less aggressive, fLEDBAT manages to distribute the available resources more equally among flows. When we are not in the sub-packet regime, LEDBAT fairness is significantly low due to the “late-comer advantage” that was discussed in Section II C. fLEDBAT was proposed as a solution to the “late-comer advantage” and, as seen in Fig. 2, it indeed solves this problem. When we enter the sub-packet regime, the network is full and packets from all flows are dropped. Therefore, the “late-comer advantage” is not present in the sub-packet regime. Another important observation from Fig. 2(a) is that fairness significantly decreases as the number of flows increases. This is the result of a large number of flows sharing a small bandwidth-delay product link; small buffers cannot hold enough packets from all flows, thus resulting in significant unfairness among flows [39]. The results are similar when we use RED as an active queue management algorithm and are thus omitted. RED also solves the “late-comer advantage” problem of LEDBAT.

3) Average packet loss probability

We investigate the average packet loss probability for each scenario by increasing traffic load, as depicted in Fig. 3 and Fig. 4 for DropTail and RED, respectively. In Fig. 3, when we
are not in the sub-packet regime, the results show that TCP flows encounter higher packet loss probability than LEDBAT and fLEDBAT in all scenarios, since TCP flows produce significantly more data. When we are in the sub-packet regime, the total data rate produced by all flows surpasses the capacity of the link, leading to increased packet loss probability. All access methods converge towards the same average packet loss probability for increasing number of flows, since flows transmit more and more packets that the buffers cannot store. This convergence is more gradual for scenarios (b) and (c), where the bandwidth-delay product is considerable and buffer sizes are larger. In all cases, given the small buffer sizes and the sub-packet regime, packet loss probability is significant. All simulations were repeated using RED as depicted in Fig. 4. As expected, in this case the average packet loss probability is higher, since RED drops packets even if the buffer is not full yet. Due to the small buffer size in scenario (a), there is no significant difference between Fig. 3(a) and Fig. 4(a).

4) Average queuing delay index

We study how the increasing traffic load in the uplink affects the average queuing delay of the flows. Fig. 5 depicts the average queuing delay index when we use DropTail. When we are not in the sub-packet regime, in all cases LEDBAT achieves significantly less queuing delay than TCP. When we enter the sub-packet regime, all access methods converge to the same average queuing delay. Another important observation is the fact that in Fig. 5(a), the maximum queuing delay index is never reached. As explained in [39], if the bottleneck buffer is not large enough to accommodate an identical number of packets from all competing flows, there are difficulties in measuring the equilibrium queuing delay. Fig. 5(b) and Fig. 5(c) that satisfy the aforementioned requirement, reach the maximum queuing delay index.

The simulations were repeated using RED in Fig. 6. Compared to Fig. 5, we notice that in all cases the average queuing delay index is significantly lower when RED is used, since buffer capacity is not fully utilized, rather packets are dropped even when the buffer is not full. The average queuing delay index in Fig. 6 is dependent on RED minimum and maximum thresholds. When we are not in the sub-packet regime, NewReno presents slightly higher average queuing delay. Using RED in the sub-packet regime, all access methods present the same average queuing delay.

B. Distribution of Resources when TCP and fLEDBAT flows share the same link

In the second part of our analysis, we consider an increasing number of TCP and fLEDBAT flows that share the same link. The aim of these scenarios is to investigate whether fLEDBAT satisfies its design principles by yielding to TCP flows in the sub-packet regime. For this reason, we study the load distribution between TCP and fLEDBAT flows in scenarios (a)-(c) for increasing traffic load using DropTail, as depicted in Fig. 7. It is noted that the number of flows in Fig. 7 and Fig. 8 refers to the total number of flows, equally divided into NewReno and fLEDBAT flows. Intuitively, we would expect a high TCP share and a low LEDBAT share. This holds only for Fig. 7(c), where both buffer size and bandwidth are significant. Due to the small buffer size and the restricted resources available in Fig. 7(a), we notice that even a few fLEDBAT flows consume a significant part of resources.
Moreover, Fig. 7(a) shows that as the number of total flows increases, fLEDBAT flows become more aggressive, consuming almost equal share of resources to TCP. Even worse, in Fig. 7(b), fLEDBAT flows become extremely aggressive, consuming even more resources than TCP when more than 80 flows share the link. This aggressiveness is the result of the incorrect base delay estimation of fLEDBAT, due to the standing buffer queue. In essence, fLEDBAT flows that enter the network when we are already in the sub-packet regime do not measure the actual base delay (i.e. when the buffer is empty), but the one-way delay when the buffer is already full. Therefore, these flows assume that there is room to increase their congestion window, becoming very aggressive when they should not. This incorrect base delay estimation [12] is obvious in all cases, where the load share of fLEDBAT gradually increases.
All simulations were repeated using RED for active queue management and the results are depicted in Fig. 8. Due to the small buffer sizes in scenario (a), no significant change in the load distribution was observed between Fig. 7(a) and Fig. 8(a). The packets drops induced by RED in Fig. 8(b) and Fig. 8(c), result in fluctuations in the one-way delay measured by iLEDBAT flows. This delay variability is misinterpreted and iLEDBAT flows constantly increase their congestion window, becoming more and more aggressive. We see that iLEDBAT flows consume more resources than NewReno flows for 50 flows in scenario (b) (Fig. 8(b)) or 80 flows in scenario (c) (Fig. 8(c)). The fact that AQM totally jeopardizes the mechanisms of scavenger transport methods has been first proposed in [38].

V. DISCUSSION AND RECOMMENDATIONS

The motivation behind this work was to investigate the suitability of a scavenger access method, such as LEDBAT, as an access method for backhaul links of wireless community networks in developing regions, where a low-bandwidth link is usually shared among a large user base, resulting in sub-packet regimes. Based on the simulation results in Section IV, we show that while out of the sub-packet regime LEDBAT is more conservative than TCP, thus underutilizing the available resources, when we enter the sub-packet regime LEDBAT presents less retransmissions achieving higher link efficiency. Moreover, iLEDBAT performs better resource distribution among its flows compared to NewReno, achieving increased fairness in all cases. If the link buffers are large enough to accommodate packets from all flows, LEDBAT also achieves lower packet loss probability.

In Section IV B, we showed that when we are not in the sub-packet regime, iLEDBAT satisfies its design principles and yields to TCP flows. When NewReno and iLEDBAT flows share the same link in the sub-packet regime, iLEDBAT flows fail to measure the actual base delay due to the standing queue and become aggressive, consuming more and more resources. In order for LEDBAT to function properly in the sub-packet regime when competing with TCP flows, new ways to estimate base delay need to be developed. Shared bottleneck detection mechanisms have been proposed in literature [40], however no real-life validation has taken place so far. Moreover, a conservative reaction to consecutive timeouts, which are typical in the sub-packet regime, needs to be incorporated in LEDBAT.

All simulations described in Section IV were also performed using different target and ζ parameter values. We have concluded that, when in the sub-packet regime, target and ζ parameters have no impact on the performance of iLEDBAT. All simulations were also performed for increased buffer sizes. Large buffers can accommodate more packets from different senders resulting in higher fairness between LEDBAT flows and significantly lower packet drop probability in the sub-packet regime. Link is fully utilized and, as expected, average queuing delay increases. When iLEDBAT flows share the same link with NewReno flows, larger buffer sizes slightly mitigate the aggressiveness of iLEDBAT. Simulations were also repeated for short FTP flows that send different file sizes. Given the nature of the sub-packet regime, where the link is always occupied, the behavior of all access methods was identical to long-lived FTP flows.

VI. CONCLUSION

Internet access is crucial. However, more than 5 billion people are without Internet access. Although there have been several initiatives in the past trying to tackle this problem, the notion of enabling the other 5 billion has gained prevalence recently with the emergence of the Alliance for Affordable Internet [41], Internet.org [42] etc. There are several ways of tackling the problem of solving access challenges: constructing new longer-range low cost wireless infrastructures, satellite access etc. There have been alternate approaches like the Access Wi-Fi Service (PAWS) [43][44], which uses Wi-Fi crowd-sharing, where existing Internet users share their home broadband connections with the poor for free. Even though Internet access infrastructures can be set up, backhaul Internet capacity is always a costly resource especially for areas with deprived connectivity [45]. Recently Facebook’s founder Mark Zuckerberg has been pressing for compression of transmitted web data to reduce costly data usage [46].

The ubiquitous nature of cloud-centric applications and user-generated content imposes a serious challenge to the underprivileged population who have limited access to costly Internet backhaul capacity. Such content, which does not have any strict real-time requirements, consumes a significant amount of the costly and, hence, precious Internet backhaul capacity. This paper aims to address the problem imposed by such applications by enabling them to use scavenger transport methods instead of the traditional TCP transport methods. Such methods will enable the applications to be transmitted without impacting other competing (real-time and, hence, more important) flows, efficiently utilising the network capacity.

Our findings show that LEDBAT can be the scavenger solution to this problem, since it achieves high link efficiency, fairness among its flows and reduced packet loss probability. When sharing the same link with TCP flows, LEDBAT becomes aggressive in certain range of scenarios. In order to solve this problem, we plan to incorporate into LEDBAT shared bottleneck detection mechanisms and a more conservative reaction to consecutive timeouts, in order to correctly adjust its congestion window in the sub-packet regime.

ACKNOWLEDGMENT

The authors wish to thank Prof. Michael Welzl for his valuable comments on the research summarized here.

REFERENCES


[41] Alliance for affordable Internet, https://www.a4i.org


