EAI Endorsed Transactions
on Cloud Systems
Research Article ICST.ORG

Multipath Bandwidth Scavenging in the Internet of Things[★]

Isabel Montes[1], Romel Parmis[1], Roel Ocampo [1], Cedric Festin [2]

[1]Computer Networks Laboratory, Electrical and Electronics Engineering Institute, University of the Philipines Diliman
[2]Networks and Distributed Systems Group, Department of Computer Science, University of the Philippines Diliman

Abstract

To meet the infrastructure coverage and capacity needed by future IoT applications, service providers may engage in mutually-beneficial modes of collaboration such as cooperative packet forwarding and gatewaying through fixed backhauls and Internet uplinks. In an effort to enable these modes of resource pooling while minimizing negative impact on collaborating providers, we developed a transport-layer approach that would enable IoT nodes to opportunistically scavenge for idle bandwidth across multiple paths. Our approach combines multipath techniques with less-than-best effort (LBE) congestion control methods. Initial tests using the TCP-LP and LEDBAT LBE algorithms on scavenging secondary flows show that this desired functionality can be achieved. To ensure however that IoT nodes are guaranteed at least one flow that fairly competes for fair share of network capacity, one flow called the primary flow uses standard TCP congestion control.

Keywords: Internet of Things, bandwidth scavenging, less-than-best-effort, congestion control, multipath flows

Received on 02 December 2014, accepted on 20 January 2015, published on 23 February 2015

doi: 10.4108/cs.1.1.e3

1. Introduction

As the Internet of Things (IoT) continues to evolve, service provides will face new challenges in the provisioning of connectivity requirements, including fixed gateways and backhauls to cloud services for the aggregation, processing, storage and distribution of data obtained from smart objects and devices. Such gateways and backhauls must be engineered to guarantee acceptable service levels given aggregate traffic volumes from a large number of sources of data. These data sources may be spread over large geographic areas, and may even be mobile. These challenges are further exacerbated by the need to strategically locate access points and gateways in a manner that would minimize energy-consuming packet forwarding within the wireless network of objects.

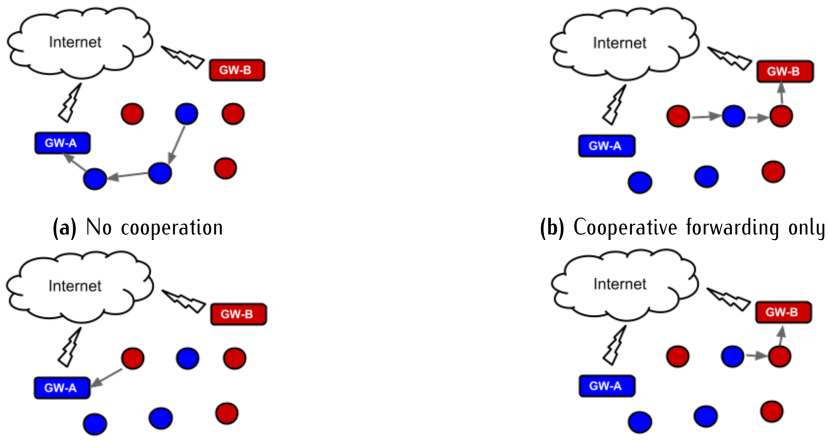
These design challenges will impose significant capital and operating costs to future IoT service providers.To complement long term efforts to engineer for maximum geographic coverage and peak traffic loads, IoT providers servicing overlapping areas may consider mutually-beneficial bilateral commercial agreements enabling transit and cooperative access through their peers' infrastructure and nodes. Such inter-IoT provider cooperation would face several design and implementation challenges, as discussed in the next section

1.1. IoT Service Provider Cooperation: Models and Challenges

Consider two IoT service providers A and B, each with respective member-nodes (smart objects, mobile devices and others) and infrastructure (fixed gateways and backhaul links). Figure 1 depicts three possible models of cooperation along with the default 'no-cooperation' scenario. These are:

- **No cooperation** [Fig. 1a]. Packets from A's nodes may only be forwarded through peer nodes from A, and gatewayed to A's infrastructure

[★]This paper is an extended version of [1]. We have added a new section (1.1) that discusses models of IoT service provider cooperation, included additional results, and have revised the rest of the paper to further incorporate useful feedback obtained from reviewers and from the conference



(a) No cooperation

(b) Cooperative forwarding only

(c) Cooperative gatewaying only

(d) Cooperative forwarding and gatewaying

Figure 1. Various IoT service provider cooperative schemes are illustrated in (b)–(d), while the default 'no-cooperation' scenario is depicted in (a) above.

- **Cooperative forwarding only** [Fig. 1b]. Packets from B may be forwarded by nodes of both A and B, but may only be gatewayed via B's infrastructure. This effectively extends the footprint of provider B without having to deploy additional gateways. Provided that A and B have agreed to settlement-free mutual packet forwarding, or otherwise have appropriate (though possibly resource-intensive) accounting mechanisms in place within wireless nodes, there will be no need for Provider A's gateways to account for B's traffic at the gateway.

- **Cooperative gatewaying only** [Fig. 1c]. Packets from B may only be forwarded by peer nodes from B, but may be gatewayed to either A or B's infrastructure. This scheme prevents resources and energy in A's nodes from being consumed by the task of forwarding traffic from B. However, Provider B's coverage is still enhanced by the availability of additional gateways and backhauls from Provider A. Provider B can optimally balance the savings from potentially shortened forwarding path lengths within its wireless network by shunting them to A's gateways with the financial cost of using A's infrastructure in the process. Additionally, it may be more practical to account for usage in gateways than in the nodes themselves (as what might have to be done in cooperative forwarding).

- **Cooperative forwarding and gatewaying** [Fig. 1d]. This is the most flexible form of cooperation, which allows both providers to cooperatively forward packets through each other's nodes and infrastructure. However, it can also be the most challenging approach in terms of usage accounting and ensuring QoS.

Any of the cooperative scenarios above may be further enhanced if nodes may concurrently exploit multiple paths through the additional resources of cooperating providers. As a simple example, suppose non-interfering gateways from Provider A and Provider B are both within range of a node from A. In a cooperative gatewaying scheme, node A can potentially benefit only if it is able to concurrently transmit to both gateways at an effective aggregate rate greater than the rate available via either gateway alone.

1.2. Multipath Bandwidth Scavenging

In both cooperative and non-cooperative scenarios, although current routing techniques allow packets from a single flow to be forwarded across different paths and gateways, naively striping packets onto multiple paths may cause problems for transport layer protocols with congestion control functionality and reliable in-order delivery. Heterogeneous path delays and loss characteristics may trigger timeouts and retransmissions, as well as head-of-line blocking at receiver buffers, forcing larger and longer buffering to be done [2–4]. A better alternative would be to partition application flows into subflows and enforce per-subflow congestion control and reliability mechanisms that adapt and respond to per-subflow path congestion and loss events [5]. Maintaining TCP-like flow semantics within individual subflows also yields better compatibility with stateful middleboxes [6]. These have been the general strategies taken by the Internet community with Multipath TCP (MPTCP) which is envisioned to provide TCP the capability to utilize multiple paths between source and destination for redundancy and better resource usage [5, 6].

While MPTCP can provide the multipath capability we require, TCP's (and MPTCP's) fairness characteristics however might not exactly sit well with competing providers who are primarily concerned with the SLAs of their own customers. Providers may not wish to fairly share bandwidth with competitors, especially when the latter merely want to opportunistically exploit bandwidth resources on top of what they already have within their own networks. In contrast, an IoT service provider might only allow a competitor to scavenge whatever remaining available bandwidth, if any, is available.

Although it may be relatively straightforward to impose differentiated QoS treatment at gateways, within the wireless network itself, a per-hop QoS approach that involves traffic classification, the management of multiple queues, and the enforcement of differentiated QoS policies might be too resource- and energy-intensive. Thus, alternative mechanisms to protect primary subflows, or flows that originate from a provider's own nodes (analogous to traffic from primary users in the context of spectrum whitespace usage by cognitive radios [7]), possibly through endpoint rather than per-hop behavior, need to be devised.

1.3. Less-Than-Best Effort Congestion Control as a Scavenging Mechanism

While we wish to have an ability to exploit and use multiple paths, opportunistic scavenging secondary subflows that are too aggressive may negatively impact the ability of other nodes to use the network. In scavenging scenarios, a paramount concern is to minimize negative impact on entities volunteering the use of idle resources [8]. This makes TCP's fairness incompatible with opportunistic endpoint-based scavenging behavior, for the following reasons:

- A secondary scavenging subflow will fairly compete for bandwidth with a primary subflow, and
- In a shared wireless medium, secondary scavenging TCP subflows traversing multiple paths may compete with primary TCP subflows over relatively wide areas of the network

These may be mitigated through the use of congestion control mechanisms that detect the onset of congestion more quickly than conventional packet loss-based ones, and yield network usage to primary subflows. The less-than-best effort (LBE) class of congestion control algorithms may offer this ability through rapid congestion detection, such as through delay measurement [9]. When mixed with TCP flows in a bottleneck link, LBE flows yield bandwidth. Furthermore, an LBE flow will also attempt to maximize the use of the available bandwidth if there are no competing flows. These characteristics make LBE congestion control a good candidate mechanism for opportunistic bandwidth scavenging.

Building on current work by others on LBE congestion control and MPTCP, we developed a hybrid transport-layer approach to concurrent multipath bandwidth scavenging that combines MPTCP's multipath mechanisms with LBE congestion control. Section 2 of this paper starts with a discussion on its basic design, while Section 3 presents initial results from our effort to validate functionality and behavior. Section 4 briefly reviews related work, while Section 5 concludes and outlines future work.

2. MP-LBE Design

In MP-LBE, two communicating endpoints start by establishing a single primary subflow that uses standard TCP-like congestion control. In cooperative scenarios, we assume that underlying routing mechanisms ensure that the primary subflow's path consists of nodes and gateways of the same provider. Secondary subflows that use LBE congestion control mechanisms are then launched on any other discovered paths. These secondary subflows essentially perform bandwidth scavenging, opportunistically using its own and competing providers' resources.

In order to balance congestion among its subflows, MPTCP uses a coupled congestion control algorithm that influences the per-subflow congestion control. This way, MPTCP moves more of its traffic away from the more congested subflows, and it maintains TCP-friendliness when sharing bottleneck links with standard TCP-like traffic. Unlike MPTCP however, MP-LBE does not employ coupled congestion control because LBEs already avoid congested links by design.

2.1. Congestion Control in Secondary Subflows

We aim to achieve low-impact multipath bandwidth scavenging by exploring the use of the LBE class of congestion control methods in secondary subflows. Although there are several methods in this class, we started with a comparative evaluation of TCP-LP and LEDBAT, with a view of expanding these evaluations to other algorithms in the future.

TCP-LP. TCP-LP is a congestion control algorithm that manages the congestion window of the sender based on the one-way forward delay experienced by the traffic on a bottleneck [10]. These one-way delay (owd) measurements approximate queuing delay, and variations in these delays allow TCP-LP to infer congestion earlier than standard TCP through a simple threshold-based algorithm.

One-way Delay Calculation: Upon receiving an ACK, TCP-LP calculates owd from the difference between the receiver's timestamp in the ACK and the sender's timestamp from the original sent packet, which the receiver copies into the ACK and echoes back to the sender. A delay smoothing parameter γ prevents false early congestion indications due to large but short-term variations in network delay coming from bursty cross traffic. TCP-LP computes the exponentially weighted moving average of owd as

$$sd_i = (1-\gamma)sd_i + \gamma d_i \tag{1}$$

where d_i is the owd of the packet i and sd_i is the smoothed owd.

Delay Threshold: TCP-LP tracks the maximum owd (d_{max}) and minimum owd (d_{min}) measurements throughout the connection. Whenever owd is calculated, it is compared to last measured d_{max} and d_{min} values, which are then replaced with the new owd if needed, before calculating the smoothed owd. The d_{min} estimates propagation delay, and $d_{max} - d_{min}$ thus estimates the maximum queueing delay. When the smoothed owd exceeds the sum of the propagation delay plus a fraction of the maximum observed queueing delay on that path, congestion is inferred.

$$sd_i > d_{min} + (d_{max} - d_{min})\delta \tag{2}$$

Congestion Avoidance Policy: When congestion inferred from the threshold formula above, TCP-LP cuts the congestion window by half and enters an inference phase wherein it awaits further congestion indication until the inference phase timeout expires. If congestion is detected during the inference phase, $cwnd$ is reduced to the size of 1. Otherwise, TCP-LP proceeds with an additive increase of $cwnd$.

LEDBAT. LEDBAT is very similar to TCP-LP in that it also uses owd measurements to infer congestion. Like TCP-LP, it measures owd using the timestamps carried by the ACKs received at the sender side. In place of TCP-LP's threshold-based algorithm for inferring congestion, LEDBAT makes use of a target queuing delay value. When queuing delay becomes higher than a specified target value, LEDBAT takes this as an indication that there is a large amount of traffic piling in a buffer somewhere in the network. Congestion is then assumed and LEDBAT reduces its sending rate to alleviate the potential congestion in the network.

Queuing Delay Measurement: On a typical noiseless path, end-to-end delay is generally composed of transmission delay (d_{tr}), propagation delay (d_p), queuing delay (d_q), and processing delay (d_{pr}). All delays are assumed constant except for the queuing delay. The constant delays are measured through base delay. LEDBAT assumes that minimum owd results from a path with zero queues on its buffers. Continuous measurement of owd over a selected observation window must be done to account for route changes that can result to a change in the constant delays. In principle, the smaller the duration of the observation window, the more responsive the LEDBAT is. On the contrary, if the observation window is too large, the LEDBAT cannot account for frequent route changes. For every sampling, base delay is updated by getting the minimum between the current base delay and the owd:

$$d_{base} = min(owd, d_{base}) \tag{3}$$

Queuing delay is computed by subtracting the base delay from the owd as shown in the equations below. Since LEDBAT can approximate base delays, all other delays aside from the base delay on an owd is assumed to be queuing delay. Delay measurements must be low-pass filtered to avoid unstable values that in turn may cause erratic sending rates [11].

$$\begin{aligned} owd &\approx d + d_p + d_q + d_{tr} \\ d_{base} &\approx d + d_p + d_{tr} \\ d_q &\approx owd - d_{base} \end{aligned} \tag{4}$$

Avoiding Congestion: LEDBAT avoids congestion by regulating the $cwnd$ size using a proportional-integral-derivative (PID) controller. The controller varies the $cwnd$ proportional to the difference of the queueing delay and the target value. For every ACK received at the sender side, the $cwnd$ adjustment is calculated as

$$off_{TARGET} = (TARGET - d_q)/TARGET \tag{5}$$

$$cwnd+ = GAIN * off_{TARGET} * bytes_{newlyacked} * \frac{MSS}{cwnd} \tag{6}$$

When queuing delay is greater than the target delay, off_{TARGET} becomes a negative value, and $cwnd$ is reduced. When queueing delay is less than the target, $cwnd$ is increased.

LEDBAT normally infers congestion before loss-based TCP, thus backing off before packet loss events occur. However, if packet loss still occurs, LEDBAT must treat this loss as a strong sign of congestion. LEDBAT would then react like standard TCP where, instead of performing PID control, it does a multiplicative decrease of its $cwnd$.

2.2. Congestion Control in Primary Subflows

Our primary subflows use standard TCP SACK congestion control, with slow start, congestion avoidance, fast retransmit, and fast recovery congestion control algorithms. The congestion control mechanism is loss-based and does not detect congestion as early as the delay-based algorithms of TCP-LP and LEDBAT.

When TCP connections share a bottleneck, they react to congestion in a way that tends to divide the bottleneck link capacity evenly among all the connections. MP-LBE's primary subflows are expected to behave similarly, and in cases where there are no available links that secondary subflows can scavenge, the minimum throughput that MP-LBE should attain should be at least as much as a TCP-share of its primary subflow's link.

3. Evaluation

We modified Nishida's implementation of MPTCP for NS-2 [12] and disabled congestion control coupling between subflows. We configured the first subflow to use standard TCP congestion control, while succeeding subflows added to the connection used an LBE congestion control algorithm. We tested two versions of MP-LBE: one that used LEDBAT on secondary flows and another that used TCP-LP. Our LEDBAT implementation for MP-LBE used a target queueing delay value of 12ms, while our MP-LBE TCP-LP implementation used γ=1/8 and δ=0.25 for the MP-LBE TCP-LP implementation.

The topology used in all the simulations is shown in Figure 2. An MP-LBE connection is configured with two subflows, one primary and one secondary subflow, and each of these share a bottleneck link with a TCP connection. The bottleneck links each have a capacity of 5Mbps and 5ms delay. The link used by the primary subflow will be referred to as the top link, while the link used by the secondary subflow will be referred to as the bottom link.

3.1. Bandwidth Scavenging Behavior

We first explore MP-LBE's ability to scavenge for additional bandwidth from idle links. In this simulation, both bottleneck links have competing standard TCP traffic. MP-LBE's primary subflow should share its link with the competing traffic (TCP-fashion), while the secondary link should give way to the competing traffic. Halfway into the simulation, the TCP traffic on the bottom link ends. This frees up the bottom link, and MP-LBE's secondary flow should react by maximizing the available bandwidth once the link becomes idle.

Both MP-LBE (LEDBAT) and MP-LBE (TCP-LP) are able to achieve this behavior, as seen in Figures 3 and 4. When the secondary flow is using LEDBAT,

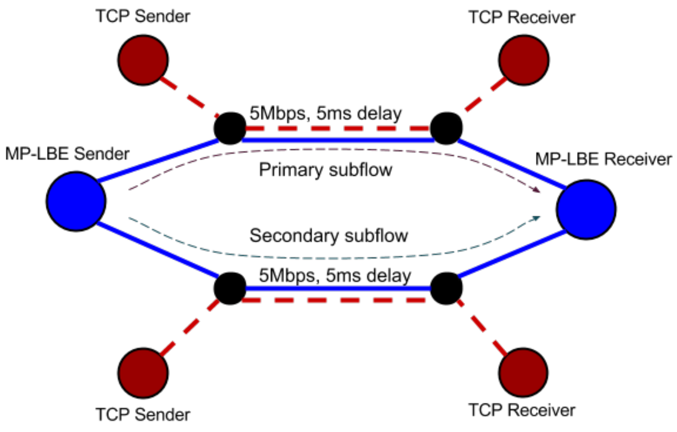


Figure 2. Simulation topology. Each access link directly connected to sender and receiver have 100Mbps capacity, with 5ms delay. Both bottleneck links have 50Mbps capacity with 5ms delay.

simulations show that it is able to maximize the available bandwidth better than TCP-LP. LEDBAT achieves a steadier throughput because its *cwnd* size does not change as drastically as that of TCP-LP.

3.2. LBE Behavior

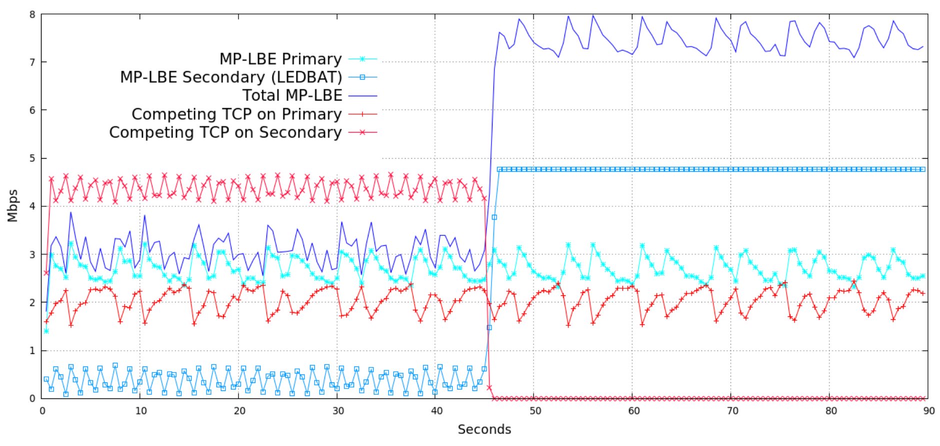
To demonstrate the LBE behavior of the secondary flow, we used the same topology, but this time only the primary flow was made to compete with regular TCP at the start of the simulation. The bottom link had no competing traffic, which allowed the secondary flow to maximize 5Mbps capacity of the link. At 45 seconds, a regular TCP connection was started on the bottom link.

Figures 5 and 6 show the simulation results. MP-LBE using TCP-LP on its secondary link was able to back off more rapidly than in the case of LEDBAT, but both demonstrated correct LBE behavior when the competing TCP traffic on the bottom link was started.

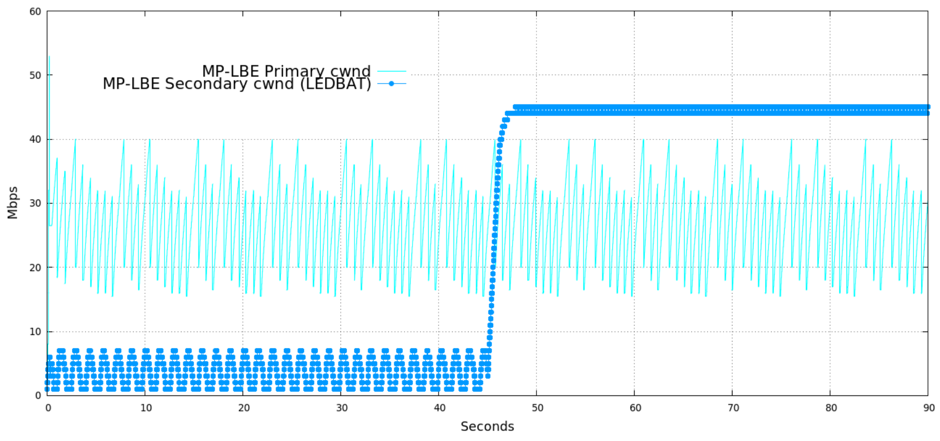
3.3. Goodput

Even though bandwidth aggregation from multiple paths improves throughput, the goodput achieved may be less than the theoretical maximum due to out-of-order arrival of packets. To evaluate MP-LBE's goodput performance, we recorded the data-level sequence numbers (DSNs) received by the destination node as a function of time.

We used the same topology as in Figure 2, but eliminated all competing TCP traffic. We ran the simulation using regular MPTCP, in addition to the simulation runs for MP-LBE (LEDBAT) and MP-LBE (TCP-LP). Figure 7 shows the data obtained from our simulation. The y-axis is scaled to 1:536, as 536 is the data-level length and sequence numbers are in increments of 536.



(a) Throughput



(b) $cwnd$ behavior

Figure 3. Bandwidth scavenging behavior of MP-LBE using LEDBAT.

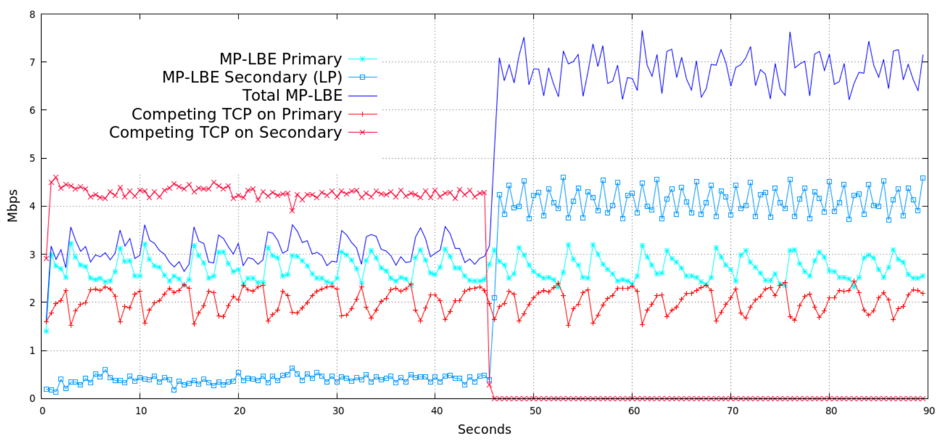
Figure 7 shows almost-identical rates of DSN increase in both MPTCP and MP-LBE (LEDBAT). MP-LBE (TCP-LP) on the other hand registers significantly lower data sequence numbers received within the same timeframe.

Figure 8 shows that during the first 3 seconds of the simulation MP-LBE's primary subflow (for both LEDBAT and TCP-LP) received sequence numbers at a slower rate than the secondary subflow. This caused goodput to suffer because only the DSNs on the secondary subflow are arriving, while all the sequence numbers sent through the primary subflow are delayed. In the case of MP-LBE (LEDBAT), the delayed packets finally arrive a little after 3 seconds and the primary subflow picks up its pace. After this point, the slope of MP-LBE (LEDBAT)'s DSN curve matches that of MPTCP.

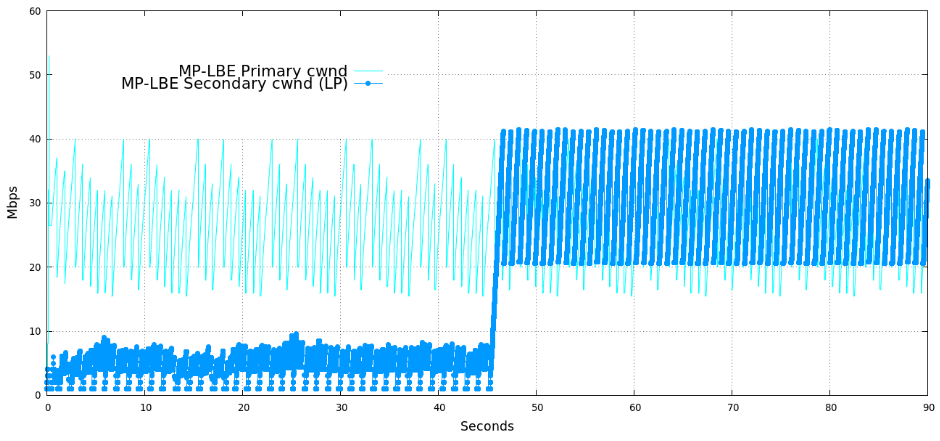
4. Related Work

Resource scavenging is not a new concept, having been previously used to harness idle computing resources to perform useful calculations for users other than the resource owner [8]. In more recent literature, bandwidth scavenging commonly refers to dynamic, opportunistic access to unused spectrum by cognitive radios [7]. Our approach is quite different in that it focuses on a solution at the transport layer through the use of LBE congestion control in a multipath fashion. While there has been some recent similar work on the development of a multipath version of LEDBAT called LEDBAT-MP [13], we are interested in the more general class of LBEs and intend to comparatively evaluate several of the representative algorithms for our intended application. Furthermore, our approach makes a crucial distinction between primary and secondary flows, ensuring that nodes can rely on at least one flow, the primary one, to compete fairly within the network.

In order to achieve cooperative gatewaying among providers, we need mechanisms to enable concurrent access to their respective fixed wireless infrastructure. BeWifi [14], a service rolled out by service provider Telefonica, allows users to use idle capacity through neighbors' access points within range. While BeWifi



(a) Throughput



(b) *cwnd* behavior

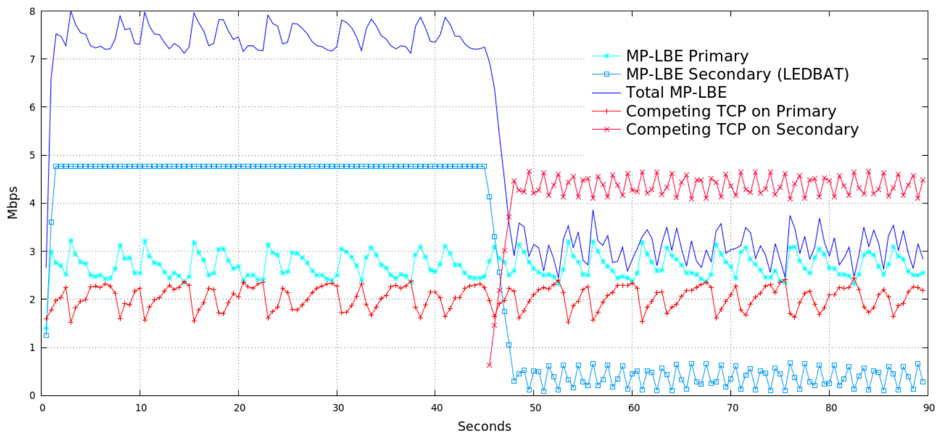
Figure 4. Bandwidth scavenging behavior of MP-LBE using TCP-LP.

applies to a single-provider model, it offers insight into the usefulness of the ability to scavenge idle bandwidth from cooperating peers. On the other hand, CableWiFi [15] employs a multi-provider model, allowing customers from five ISPs access to the consortium's infrastructure. From a technical point of view, one mechanism that can enable cooperative gatewaying is offered by BaPu (Bunching of Access Point Uplinks) [16] is a software-based approach that employs packet overhearing, using it to pool together WiFi uplinks that are in close proximity to one another. A BaPu-Gateway AP schedules which contributing BaPu-APs will send the packets through to the receiver. BaPu-APs are also configured to prioritize the home user's traffic over any background traffic that is generated when APs act as BaPu contributors. BaPu was designed primarily for uploading user-generated content over the Internet and cannot be used for downloads.

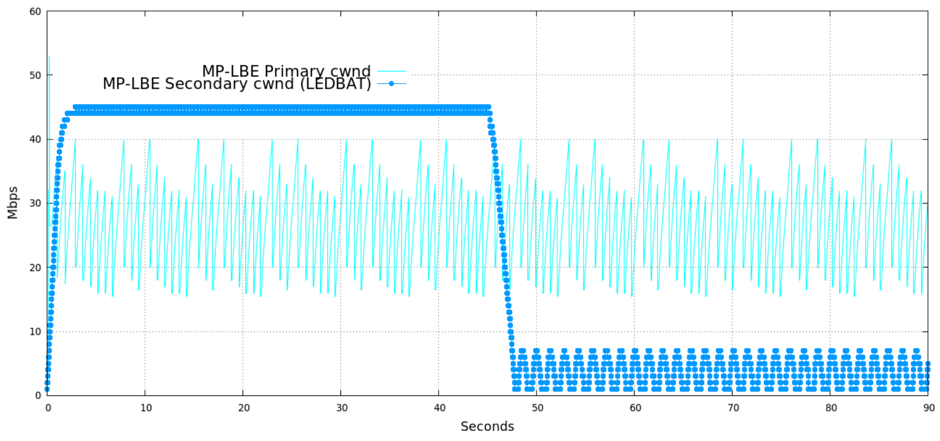
The ability to concurrently exploit multiple paths for bandwidth scavenging may also be viewed as a problem of bandwidth aggregation. Application layer solutions such as DBAS [17] typically do not require changes in the underlying protocols and instead rely on endpoint middleware to intercept traffic and manage scheduling, reordering, and transmission over multiple interfaces. DBAS' ability to deal with stateful middleboxes, as well as the ensuing fairness of its subflows is however not known. Alternatively, instead of placing the functionality within the endpoint itself, dedicated proxy middleboxes may be deployed within the network in order to do aggregation, delay equalization and packet scheduling [2, 3]. This seems to be more feasible to do within the fixed infrastructure, and represents additional cost and management overhead for providers. We preferred to take an endpoint-based approach since it offers an end-to-end solution, covering both the fixed and wireless portions of the network.

5. Conclusion and Future Work

To provide a low-impact mechanism that will encourage future IoT service providers to explore various models of cooperation, including, but not limited to,



(a) Throughput



(b) *cwnd* behavior

Figure 5. LBE Behavior of MP-LBE using LEDBAT.

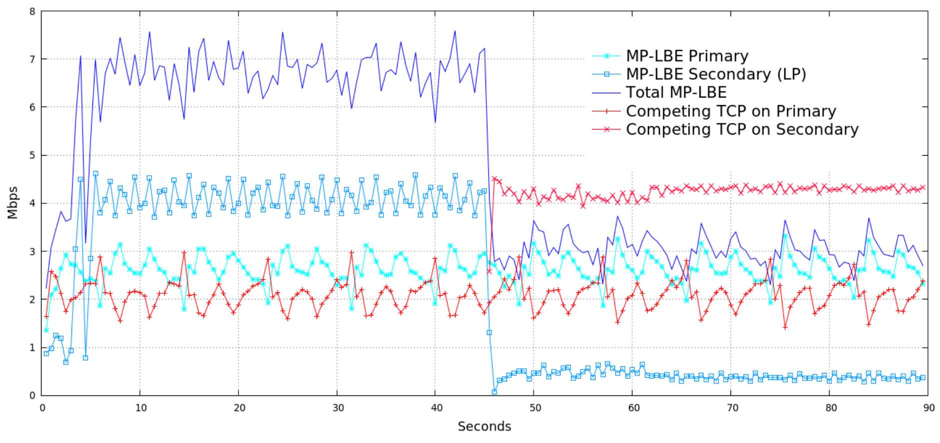
cooperative forwarding and gatewaying, we propose a transport-layer approach for multipath bandwidth scavenging that uses TCP-like congestion control for primary subflows and less-than-best effort (LBE) congestion control for secondary subflows. The use of LBE for secondary subflows ensures that these back off and yield bandwidth in the face of other traffic, including primary subflows from other IoT devices.

Our MP-LBE design effectively improves throughput when one or more idle links become available for secondary subflows. When no additional links are available, the primary subflow achieves the throughput of a single TCP flow, and secondary flows are able to rapidly use capacities along paths that become idle.

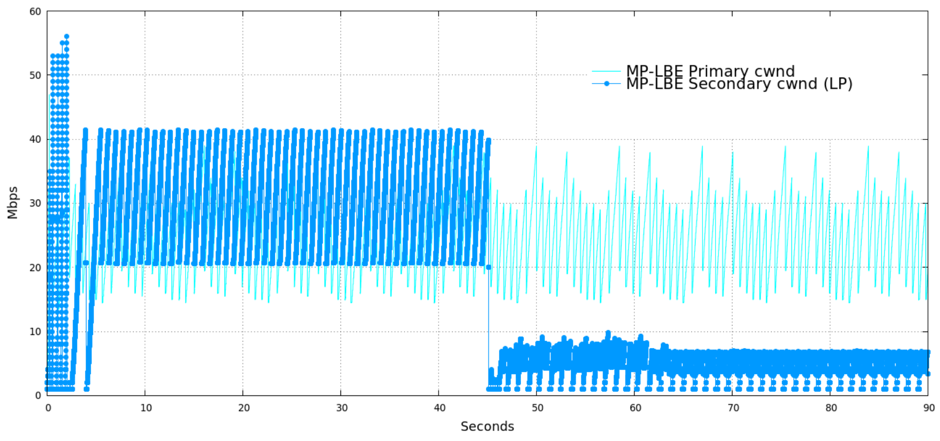
MP-LBE for both LEDBAT and TCP-LP yield lower goodput than MPTCP, with MP-LBE (LEDBAT) having worse out-of-order packet arrivals at the beginning of its connection lifetime. Noting that MPTCP employs scheduling on subflows to mitigate the impact of non-uniform path delays on packet arrivals, and consequently buffer requirements and goodput [18], we intend to work on improving goodput for MP-LBE by considering and possibly extending the various approaches that have been proposed for reducing the number of out-of-order packet arrivals in multipath connections, such as delay equalization [2], packet scheduling [3], congestion window adaptation [19].

Recognizing "less than best effort" for what it is, smart objects and devices should principally rely on primary flows to carry critical traffic. However, the ability to scavenge additional bandwidth and paths will enable IoT sensors and devices to opportunistically explore shortcut fast paths and accelerate local aggregation and processing of data, or temporarily transmit information at higher-than-fair levels of spatial and temporal resolution. Resource pooling by cooperative IoT service providers should expand these opportunities even further.

Smart objects and devices in the Internet of Things will undoubtedly dedicate most of their resources to sensing and aggregating data, and any local processing and cognitive functionality required. With any new functionality being introduced, such as multipath



(a) Throughput



(b) *cwnd* behavior

Figure 6. LBE Behavior of MP-LBE using TCP-LP.

bandwidth scavenging, prudent design dictates that there should be minimal impact on resource footprint. We intend to keep this as a guiding principle as our work moves forward.

Acknowledgement. This work has been supported by the Engineering Research and Development for Technology (ERDT) Consortium, Department of Science and Technology – Science Education Institute (DOST-SEI), Republic of the Philippines.

References

[1] Isabel Montes, Romel Parmis, Roel Ocampo, and Cedric Festin. Multipath Bandwidth Scavenging in the Internet of Things. In *Proceedings of the International Conference on Internet of Things as a Service, 2014*, 2014.

[2] Kristian Evensen, Dominik Kaspar, Paal Engelstad, Audun Fosselie Hansen, Carsten Griwodz, and Pål Halvorsen. A Network-layer Proxy for Bandwidth Aggregation and Reduction of IP Packet Reordering. In *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pages 585–592. IEEE, 2009.

[3] Kameswari Chebrolu, Bhaskaran Raman, and Ramesh R. Rao. A Network Layer Approach to Enable TCP over Multiple Interfaces. *Wirel. Netw.*, 11(5):637–650, September 2005.

[4] T. Zinner, K. Tutschku, A. Nakao, and P. Tran-Gia. Using Concurrent Multipath Transmission for Transport Virtualization: Analyzing Path Selection. In *Teletraffic Congress (ITC), 2010 22nd International*, pages 1–7, Sept 2010.

[5] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *NSDI*, volume 11, pages 8–8, 2011.

[6] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, Mark Handley, et al. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *USENIX Symposium of Networked Systems Design and Implementation (NSDI'12)*, 2012.

[7] Anthony Plummer Jr., Mahmoud Taghizadeh, and Subir Biswas. Measurement-Based Bandwidth Scavenging in Wireless Networks. *IEEE Transactions on Mobile Computing*, 11(1):19–32, 2012.

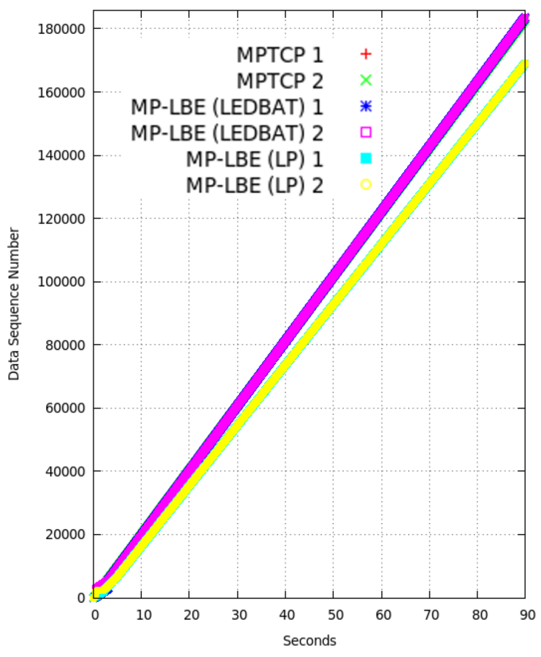


Figure 7. Data Sequence Numbers received at the destination node plotted against time.

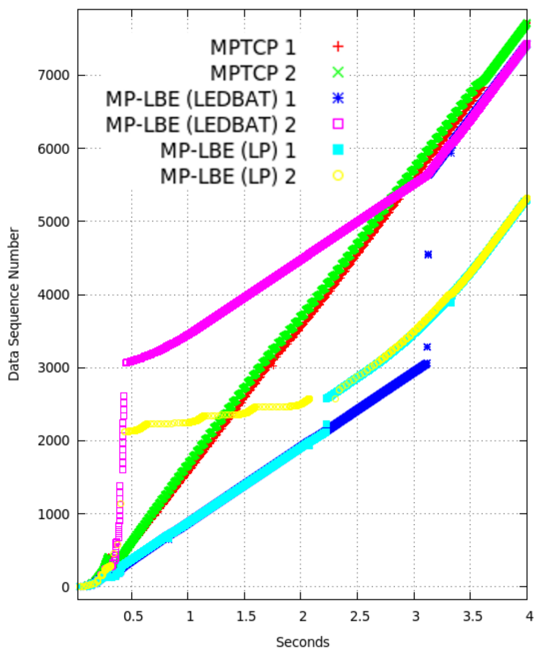


Figure 8. Data Sequence Numbers received at the destination node plotted against time, during the first 4 seconds of the simulation.

[8] Jonathan W Strickland, Vincent W Freeh, Xiaosong Ma, and Sudharshan S Vazhkudai. Governor: Autonomic Throttling for Aggressive Idle Resource Scavenging. In *Proceedings of the Second International Conference on Autonomic Computing, 2005.*, pages 64–75. IEEE, 2005.

[9] Michael Welzl and David Ros. A Survey of Lower-than-Best-Effort Transport Protocol. RFC 6297, RFC Editor, June 2011.

[10] Aleksandar Kuzmanovic and Edward W. Knightly. TCP-LP: Low-priority Service via End-point Congestion Control. *IEEE/ACM Trans. Netw.*, 14(4):739–752, August 2006.

[11] Stanislav Shalunov, Greg Hazel, Janardhan Iyengar, and Mirja Kuehlewind. Low Extra Delay Background Transport (LEDBAT). RFC 6817, RFC Editor, December 2012.

[12] Google Code Project. Multipath-TCP: Implement Multipath TCP on NS-2. http://code.google.com/p/multipath-tcp. Accessed June 30, 2014.

[13] Hakim Adhari, Sebastian Werner, Thomas Dreibholz, and Erwin Paul Rathgeb. LEDBAT-MP –On the Application of Lower-than-Best-Effort for Concurrent Multipath Transfer. In *Proceedings of the 4th International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Victoria, British Columbia/Canada, May 2014. ISBN 978-1-4799-2652-7.

[14] BeWifi. http://www.bewifi.es. Accessed June 30, 2014.

[15] Cable WiFi : Internet access brought to consumers through a collaboration among U.S. Internet Service Providers. http://www.cablewifi.com. Accessed June 30, 2014.

[16] Tao Jin, Triet Vo Huu, Erik-Oliver Blass, and Guevara Noubir. BaPu: Efficient and Practical Bunching of Access Point Uplinks. *CoRR*, abs/1301.5928, 2013.

[17] K. Habak, M. Youssef, and K.A. Harras. DBAS: A Deployable Bandwidth Aggregation System. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–6, May 2012.

[18] Christoph Paasch, Simone Ferlin, Ozgu Alay, and Olivier Bonaventure. Experimental evaluation of multipath tcp schedulers. In *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, pages 27–32. ACM, 2014.

[19] Dizhi Zhou, Wei Song, and Minghui Shi. Goodput Improvement for Multipath TCP by Congestion Window Adaptation in Multi-Radio Devices. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 508–514. IEEE, 2013.