

G-DBAS: A Green and Deployable Bandwidth Aggregation System

Karim Habak

Wireless Research Center

Egypt-Japan University of
Sc. and Tech. (E-JUST)

Email: karim.habak@ejust.edu.eg

Moustafa Youssef

Wireless Research Center

Egypt-Japan Univ. of Sc. and Tech. (E-JUST)
and Alexandria University, Alexandria, Egypt.

Email: moustafa.youssef@ejust.edu.eg

Khaled A. Harras

Computer Science Department

School of Computer Science
Carnegie Mellon University in Qatar

Email: kharras@cs.cmu.edu

Abstract—The widespread deployment of varying networking technologies, coupled with the exponential increase in end-user data demand, have all led to the proliferation of multi-homed or multi-interface enabled devices. To date, these interfaces are mainly utilized one at a time based on network availability, cost, and user-choice. Researchers have recently focused on leveraging these interfaces simultaneously by proposing solutions to aggregate their bandwidths in order to ultimately increase throughput and satisfy the end-user's growing demand on data. These solutions, however, have faced a steep deployment barrier due to various system design choices and heavy demand on energy. In this paper, we propose a novel Green and Deployable Bandwidth Aggregation System (G-DBAS) for multiple interface enabled devices. G-DBAS addresses a set of challenges including automatically estimating the characteristics of applications and scheduling various connections to different interfaces along with meeting different energy consumption goals set by users. We fully implement G-DBAS on the Windows OS and evaluate various scheduling strategies that we propose. Our implementation and simulation results show that G-DBAS can achieve the user energy-throughput goals while operating as an out-of-the-box standard Windows executable, highlighting its deployability and ease of use.

I. INTRODUCTION

With the continuous advances in wireless technologies, decreasing cost of electronics, and increased user demand for mobile data, it is the norm nowadays to find devices with multiple network interfaces. To keep up with this diversity in wireless technologies, devices such as laptops, netbooks, tablets, and various smart phones, are equipped with several network interfaces. Such devices, however, are currently incapable of fully leveraging these interfaces in order to enhance the overall system performance. Multiple interfaces can be simultaneously utilized by partitioning data across them in order to achieve higher throughput and enhance user experience.

Many approaches have addressed the multiple interface bandwidth aggregation problem by proposing various solutions at different layers of the protocol stack. Application layer solutions typically require applications to be aware of the existence of multiple interfaces and be responsible for utilizing them [1]. Socket level solutions, on the other hand, modify the kernel socket handling functions to enable existing applications in order to use multiple interfaces [1], [2]. These solutions either require application feedback making them incompatible with

previous versions [2], or require changes to legacy servers in order to support these new sockets [1]. Most bandwidth aggregation techniques, however, lie in the transport layer [3]–[12]. These solutions replace TCP with mechanisms and protocols that handle multiple interfaces. Such techniques also require changes to legacy servers and hence have a huge deployment barrier. Finally, network layer approaches update the network layer to hide the variation in interfaces from the running TCP protocol [13]–[15]. However, they require proxy servers to connect with or update the network layer protocols in both communicating end-points.

Most, if not all, of these solutions for bandwidth aggregation face a steep deployment barrier that led to our recent development of DBAS [16]. The goal of DBAS was to develop a deployable bandwidth aggregation system with a focus on maximizing the overall systems throughput. However, since battery operated mobile devices are the main target for such systems, it is crucial for these systems to be energy aware. While energy saving techniques for mobile devices have been heavily researched [17]–[21], none of them have done so in association with bandwidth aggregation by building data to interface scheduling solutions that are energy aware.

In this paper, we present a Green and Deployable Bandwidth Aggregation System (G-DBAS) for multi-interface enabled devices. Our system extends the DBAS [16] framework in order to balance overall throughput with energy consumption. We also develop various power-aware scheduling techniques where we introduce a new utility based scheduler that accommodates the tradeoff between maximizing throughput and minimizing power consumption. We evaluate G-DBAS via implementation and further validate the results via simulations. Our results show that G-DBAS leads to significantly higher throughput compared to current operating systems forced to run on a single interface at a time. More importantly, G-DBAS can be tuned by the user to obtain different energy-efficiency goals with the added advantage of being deployable.

The remainder of this paper is organized as follows. Section II presents the G-DBAS architecture that takes energy consumption into account. We then present our scheduling techniques in Section III, followed by their evaluation in Section IV. Finally, we conclude the paper in Section V.

II. A GREEN AND DEPLOYABLE BANDWIDTH AGGREGATION SYSTEM

In this section, we provide an overview of G-DBAS (Figure 1), followed by a detailed description of its components. We consider a client host, which is equipped with multiple network interfaces connected to the Internet. Each interface has its own parameters in terms of bandwidth, latency, loss ratio, and energy consumption. The device runs multiple applications with varying communication characteristics. G-DBAS schedules different connections to the interfaces such that a connection can be assigned to only one of the available interfaces. Once assigned to an interface, all the packets of this connection utilize the same interface.

Application Characteristics Estimator: To be fully deployable and make use of the applications characteristics, G-DBAS automatically estimates the characteristics of the applications based on their behavior. These characteristics are stored in a database. This functionality is the responsibility of the application characteristics estimation module that utilizes both qualitative and quantitative measures.

For qualitative measures, some features (e.g. process name) can be used to characterize the behavior of an application, its type, and requirement. For quantitative measures, on the other hand, the module also estimates the average connection data demand in bytes of any given application. After a connection is closed, the module updates the estimated values of connection data demand (C_{demand}) as:

$$C_{demand} = (1 - \alpha)C_{demand} + \alpha CC_{demand} \quad (1)$$

where CC_{demand} is the number of bytes transmitted by the closed connection and α is a smoothing coefficient, taken equal to 0.125 to evaluate the equation efficiently. We note that the granularity of estimation can be rendered more fine grained at the expense of increased complexity and lower scalability.

Interface Characteristics Estimator: This module is responsible for estimating the characteristics of each network interface. In particular, it estimates the available bandwidth at each interface [16]. It can also be used to automatically estimate the energy consumption of each interface.

To characterize energy utilization, Table I shows how the power consumption of a network interface depends on the NIC and technology used [22]. Hence, in order to make G-DBAS estimate the power consumption for its attached network interfaces, we built an online service with a database containing the power consumption rates for various network interfaces. Once G-DBAS runs, it queries the database for the power consumption rates of each network interface.

Battery Sensor: This module senses the available battery level and whether the device is plugged to a power source or

TABLE I
POWER CONSUMPTION TABLE [22]

Interface	Technology	Low-Power Idle	Active Tx
Netgear MA701	Wifi	264 mW	990 mW
Linksys WCF12	Wifi	256 mW	890 mW
BlueCore3	Bluetooth	25 mW	120 mW

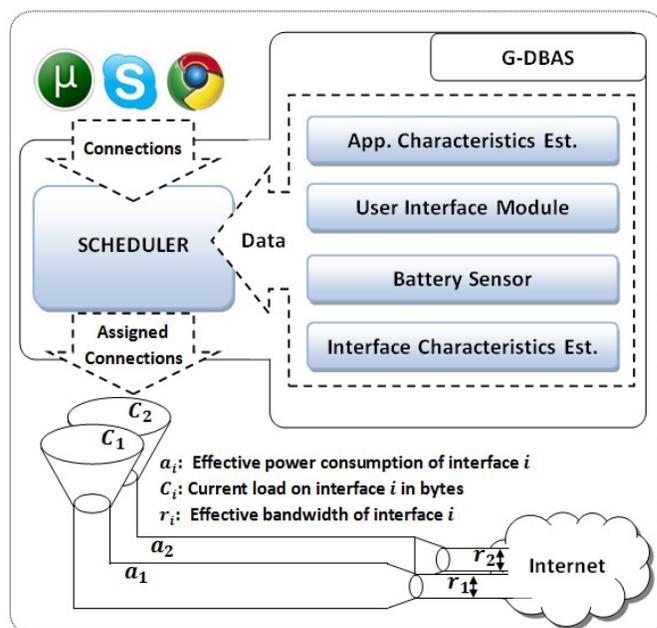


Fig. 1. G-DBAS System Architecture and Scheduling Parameters

running on battery.

User Interface Module: This module is responsible for obtaining the user's preferences and interface usage policies. The user can configure this module to enforce some interface selection criteria. For example, the user may wish to assign specific applications (e.g. realtime applications) to specific interfaces (e.g. wired). Similarly, the user might prefer to reduce cost and give higher priority/weight to interfaces that have a free connection or low power.

Scheduler: The scheduler uses the application and interface characteristics estimators; and the battery status to schedule the connections to different interfaces.

III. SCHEDULING TECHNIQUES

In this section, we present the various scheduling algorithms we adopt in G-DBAS. These algorithms are generally classified into energy unaware and energy-aware schedulers that take energy consumption rates into account when making scheduling decisions.

A. Energy Unaware Schedulers

1) *Only-One:* Used as a baseline scenario to compare against. It assigns connections to only one of the available interfaces. This assignment also represents schedulers typically used in current operating systems such as Windows or Linux.

2) *Weighted Round Robin:* Assigns connections to network interfaces in a rotating basis weighed by the interface estimated bandwidth such that higher bandwidth interfaces get assigned more connections. This scheduler shows the advantage of estimating the interfaces characteristics. On the other hand, it cannot achieve the highest performance because it does not take into account the application heterogeneity.

3) *Maximum Throughput*: This scheduler minimizes the time needed to finish the current system load in addition to the load introduced by new connections. It relies on the estimated bandwidth for each interface and the estimated connection data demand per application. It selects the interface which has the minimum T , where T is calculated as:

$$T = \frac{L + C_i}{r_i} \quad (2)$$

where L is the new connection data demand in bytes, C_i is the current load on interface i in bytes, and r_i is the effective bandwidth of interface i .

B. Energy Aware Schedulers

1) *Energy Efficient*: Connections are assigned to the network interface that will minimize the overall energy consumption. This scheduler selects the interface which minimizes E calculated as:

$$E = \frac{L}{r_i} a_i \quad (3)$$

where a_i is the energy consumption per unit time when interface i is active minus the consumption per unit time when interface i is idle.

2) *Utility-Based Scheduler*: This scheduler combines the maximum throughput and energy efficient schedulers using a factor ω that has values in the range $[0, 1]$ that would balance between losing throughput in order to minimize energy consumption. This scheduler selects the interface that minimizes \mathcal{C} calculated as:

$$\begin{aligned} \mathcal{C} &= \omega \frac{L + C_i}{r_i} \div \text{Max}_j \left(\frac{L + C_j}{r_j} \right) \\ &+ (1 - \omega) \frac{L}{r_i} a_i \div \text{Max}_j \left(\frac{L}{r_j} a_j \right) \\ &= \omega \frac{L + C_i}{r_i} \div \text{Max}_j \left(\frac{L + C_j}{r_j} \right) \\ &+ (1 - \omega) \frac{a_i}{r_i} \div \text{Max}_j \left(\frac{a_j}{r_j} \right) \end{aligned} \quad (4)$$

The idea behind this utility-based scheduler is to provide a tradeoff between the energy consumption and the overall system throughput. This tradeoff is managed by a utility function that outputs the value ω which is used by the scheduler. When $\omega = 1$, the scheduler runs in maximum throughput mode and Equation 4 will be $\frac{L+C_i}{r_i} \div \text{Max}_j \left(\frac{L+C_j}{r_j} \right)$. On the other hand, when $\omega = 0$, the scheduler makes its priority energy efficiency rendering the equation to be $\frac{a_i}{r_i} \div \text{Max}_j \left(\frac{a_j}{r_j} \right)$.

We divide each term by its maximum value over all the interfaces to normalize both terms to a value in the range $[0, 1]$.

We also note that, since the power consumption terms in the equation are constants independent from the system load, this leads to regions where certain interfaces will be completely inactive for certain values of ω . Therefore, ω has to be selected in the effective range that enables all interfaces to obtain the desired balance. This effective range can be easily obtained

based on the values of the a_j and r_j parameters as we show later in Section IV.

Utility Function: The utility function is considered the heart of the utility-based scheduler. It outputs the value ω used in Equation 4. This utility function reflects the value the user specifies for energy-consumption. It can be automatically selected based on the remaining battery level and required performance, or manually through the user interface module. We provide examples on the utility functions and their effect on performance in Section IV.

IV. PERFORMANCE EVALUATION

In this section we evaluate the performance of G-DBAS via implementation. These results have been validated using NS2 simulations but not included for space constraints. Energy consumption-related results were mainly obtained via NS2 simulations. We start by reviewing the implementation then describing our experimental setup followed by presenting and analyzing our results.

A. Implementation

G-DBAS extends the DBAS implementation [16] on the Microsoft Windows Operating System to handle energy aspects. Currently, we have clients for Windows 7, Vista, and XP. We implement G-DBAS as a middleware and a monitoring application.

We implement the G-DBAS middleware as a Layered Service Provider (LSP) [23], which is installed as part of the TCP/IP protocol stack in the Windows OS. Our G-DBAS LSP intercepts the Winsock 2 API requests and schedules the connection to its selected interface. In addition, it performs the functionality of the mode detection, application characteristic estimation, interface characteristic estimation, as well as the battery sensing described in Section II. A monitoring application, that represents the user interface module, is also implemented to capture the user's preferences and interface usage policies. It also allows users to monitor G-DBAS internal data structures and estimated values by interfacing with the G-DBAS middleware.

B. Experimental Setup

Without loss of generality, our testbed consists of three nodes: a server, a client, and an intermediate traffic shaper node. The server is the connection destination. The intermediate node is a device running the NIST-NET [24] network emulator to emulate the varying network characteristics of each interface. The client is the connection generator enabled with multiple network interfaces. On the client, we run different applications that vary in terms of the number of connections per second they open (β) and the average connection data demand (λ). The client is connected to the intermediate node through two interfaces: IF₁ and IF₂. The server is connected to the intermediate node using a single high bandwidth link L_1 . We note that the combined bandwidth of IF₁ and IF₂ is less than the server bandwidth in order to test the true impact of varying the interface characteristics and scheduling strategies.

We evaluate G-DBAS using two classes of applications: small load, and large load. Small load applications represent typical web browsing applications with an average connection data demand of $\lambda_{small} = 22.38KB$ [25]. Large load applications represent P2P and FTP applications with an average connection data demand of $\lambda_{large} = 0.285MB$ [25]. The connection establishment rate follows a poisson process with mean (β) connections per second. β_{small} , and β_{large} are changed to achieve different application mixes. Each experiment represents the average of 15 runs.

Throughout our evaluation, we use throughput and energy consumption per unit data as our metrics while varying several parameters including interfaces characteristics, and the utility function. Table II lists the parameters employed in our evaluation.

C. Results

We start by studying the impact of interface heterogeneity and application heterogeneity on the performance of the schedulers we propose. We then study the utility-based scheduler and finally discuss the effect of different utility functions and compare their performance.

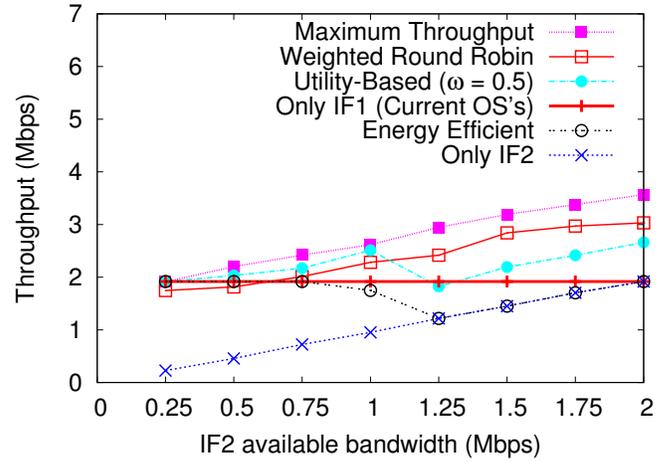
1) *Impact of Interface Heterogeneity:* In this experiment, we fix the bandwidth of IF₁ at 2Mbps and change the bandwidth of IF₂ from 0.25 to 2 Mbps. The application mix is the default as in Table II. Figure 3 shows the results. *Note that IF₂ consumes half the energy of IF₁.*

Figure 2(a) shows that the maximum throughput and weighted round robin schedulers both exploit the heterogeneity of the interfaces to achieve high throughput. For low values of the IF₂ bandwidth, using only IF₁ outperforms the weighted round robin scheduler. This is because the weighted round robin scheduler does not take the connection's load and characteristics in its decision. Therefore, long-lived connections assigned to the slow interface negatively affect the performance. This is not the case with the maximum throughput scheduler.

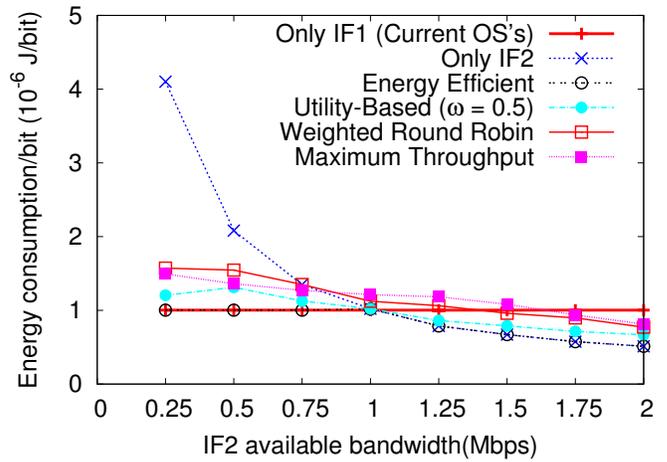
The energy efficient scheduler starts by using IF₁ only as it has the minimum energy consumption and switches completely to using IF₂ when its bandwidth becomes higher leading to less energy consumption per unit data. When both interfaces consume the same amount of energy, the scheduler uses both of them (between bandwidth = 0.75 and 1.25). The Utility-based scheduler with $\omega = 0.5$ achieves higher throughput compared to the energy efficient scheduler and lower than the maximum throughput scheduler because it combines both schedulers and has a tradeoff based on user utility

TABLE II
EXPERIMENTS PARAMETERS

Parameter	Value range	Nominal Value
L_1 (Server) Bandwidth(Mbps)	6	6
IF ₁ Bandwidth(Mbps)	2	2
IF ₁ energy consumption (Watt)	2	2
IF ₂ Bandwidth(Mbps)	0.25 - 2	2
IF ₂ energy consumption (Watt)	1	1
β_{small} (Connections/sec)	13	13
β_{large} (Connections/sec)	1	1



(a) Throughput

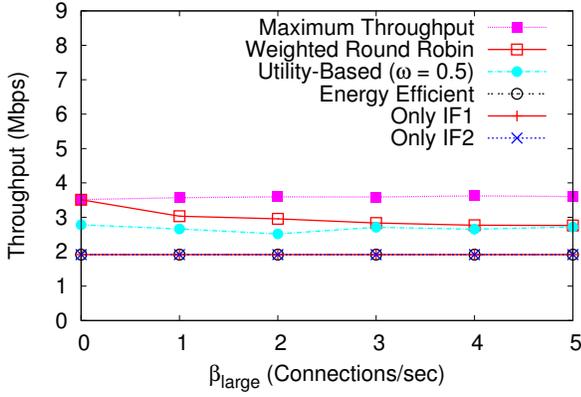


(b) Energy consumption per unit data

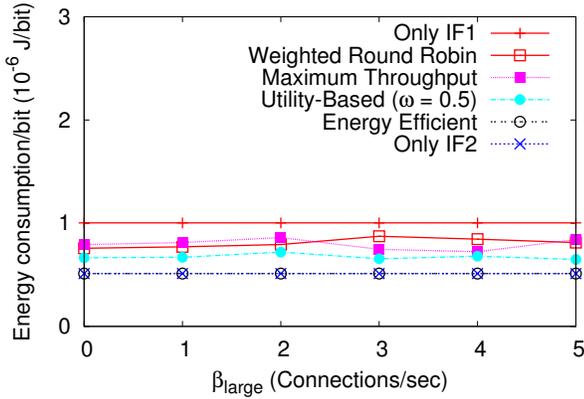
Fig. 2. Impact of interfaces heterogeneity (bandwidth).

(more in Subsection 4). For the energy efficient and utility-based schedulers, we notice a sudden throughput degradation after the bandwidth of IF₂ exceeds 1 Mbps. After this point IF₂ becomes better in terms of energy consumption than IF₁, which has a higher bandwidth.

Figure 2(b) compares the different schedulers in terms of energy consumption per unit data. The figure shows that as the bandwidth of IF₂ increases, it becomes more attractive in terms of the energy consumed per bit. Therefore, the energy efficient scheduler switches to it when its throughput reaches 1Mbps, achieving the lowest energy consumption per unit data among all schedulers. The maximum throughput and weighted round robin schedulers consume the largest amount of energy per unit data. However, they are more efficient than using a single interface for higher values of IF₂ bandwidth as they leverage the two interfaces to finish earlier, consuming less energy per bit. We note that achieving the minimum energy consumption per unit time does not lead to optimal energy consumption as it does not take the different interfaces bandwidths into account. This is why the energy efficient scheduler does not use IF₂ when its bandwidth is low.



(a) Throughput

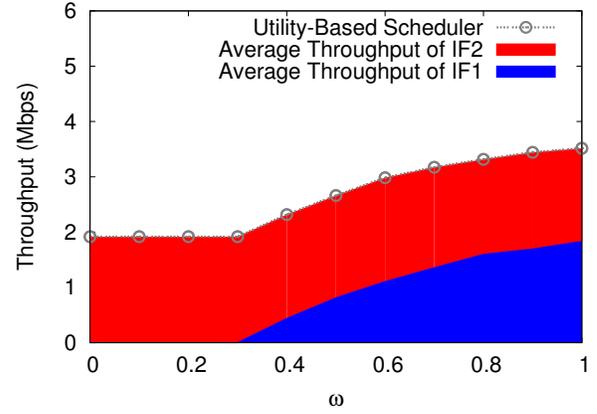


(b) Energy consumption per unit data

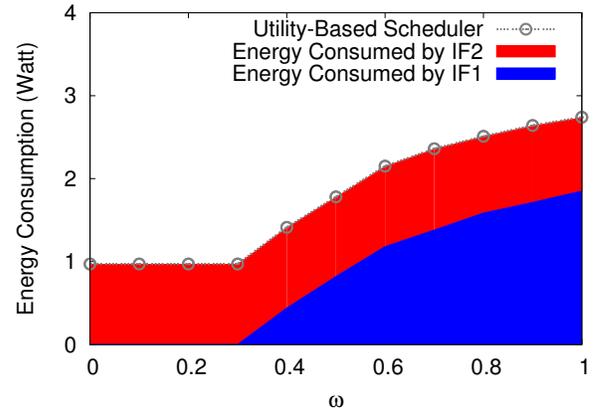
Fig. 3. Impact of applications heterogeneity.

2) *Impact of Application Heterogeneity*: Figure 3 shows the impact of changing the system workload on throughput. This experiment shows the effect of changing β_{large} . The two network interfaces are set to have a 2Mbps bandwidth. The figure shows that weighted round robin achieves a lower throughput than maximum throughput because it does not take into account the application connection characteristics. The energy efficient scheduler selects only the interface with the minimum energy consumption in order to transmit its data (since both interfaces have the same bandwidth). The utility-based scheduler takes the applications characteristics into account because part of its decision metric is the maximum throughput metric.

3) *Evaluation of the Utility-Based Scheduler*: Figure 4 shows the effect of changing ω on the performance of the system. Both interfaces are set to 2 Mbps. When $\omega = 0$, the energy efficient scheduler is dominating and always selects IF₂ as it has the minimum energy consumption (though the minimum throughput). At the other extreme, when $\omega = 1$, the maximum throughput scheduler dominates and achieves maximum throughput (sum of the two interfaces minus overhead). In between, based on the user's utility, the utility-based scheduler achieves an arbitrary throughput between both extremes. As mentioned in Section III-B2, the figure shows



(a) Throughput



(b) Energy consumption per unit time

Fig. 4. Throughput-energy tradeoff for the utility-based scheduler.

that the effective region for ω in this case is $\omega \in [0.33, 1.0]$. Outside this range, Only interface 2 is active.

4) *Evaluation of Different Utility Functions*: Figure 5 compares two different utility functions: (1) U_1 represents the case when the user's weight of energy consumption increases linearly with time. This can be the case when the utility is proportional to the remaining battery level. (2) U_2 represents the case when there is a point where the user's value of the energy consumption suddenly switches. This can be the case when the user sets a threshold on the minimum battery level for high performance after which he switches to a energy-saving mode.

The figure shows that the utility functions capture user intent: The linear utility function has both smooth changes in throughput and energy consumption (slope of the curves). The step utility function changes both the throughput and energy consumption at the change point ($t = 5$ min).

D. Discussion

Although G-DBAS is based on the DBAS system, it only uses its connection-oriented mode. This is because G-DBAS targets building a stand-alone device that does not require support from the other end; Most of the available solutions for the bandwidth aggregation problem work in the packet-

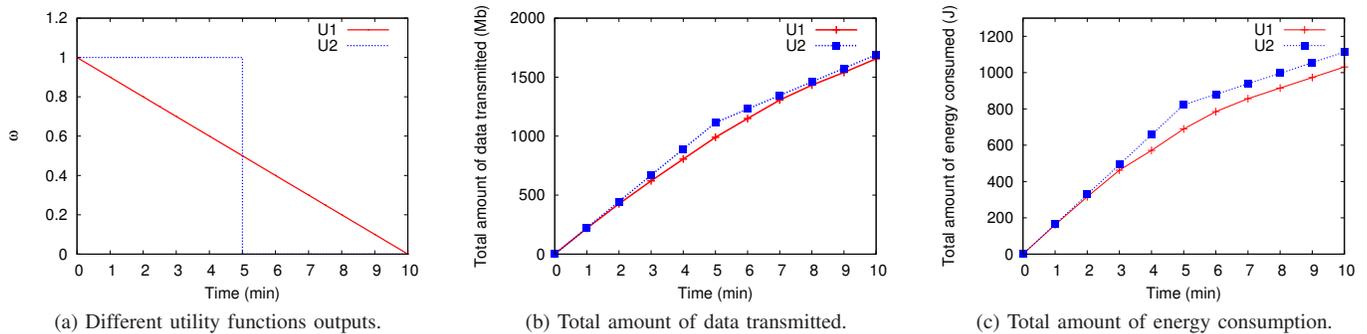


Fig. 4. Tradeoff between throughput and energy consumption.

oriented mode of scheduling to leverage its advantage of higher performance. However, due to the increase in the number of concurrent connections required by applications, G-DBAS connection-oriented scheduling can achieve comparable performance to packet-based scheduling without any changes to the current devices.

V. CONCLUSION

In this paper we have presented G-DBAS, an energy-aware deployable bandwidth aggregation system for devices with multiple network interfaces. G-DBAS presents a deployable bandwidth aggregation system that targets achieving different user goals in terms of the throughput and energy consumption. We have presented the components of our system, demonstrated how it adapts to different user energy-efficiency goals, and evaluated its performance both via implementation and simulation. Our results have shown that G-DBAS leads to significantly higher throughput compared to current operating systems. More importantly, the performance of G-DBAS can be tuned by the user to obtain the desired energy-efficiency tradeoff.

REFERENCES

- [1] H. Sakakibara, M. Saito, and H. Tokuda. Design and implementation of a socket-level bandwidth aggregation mechanism for wireless networks. In *Proceedings of the 2nd annual international workshop on Wireless internet*, page 11. ACM, 2006.
- [2] B.D. Higgins, A. Reda, T. Alperovich, J. Flinn, TJ Giuli, B. Noble, and D. Watson. Intentional networking: opportunistic exploitation of mobile network diversity. In *ACM Mobicom'10*, pages 73–84.
- [3] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. *Urbana*, 51:61801.
- [4] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *USENIX Annual Technical Conference*, 2004.
- [5] H.Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1):99–114, 2005.
- [6] Y. Dong, D. Wang, N. Pissinou, and J. Wang. Multi-path load balancing in transport layer. In *Next Generation Internet Networks, 3rd EuroNGI Conference on*, pages 135–142. IEEE, 2007.
- [7] A. Argyriou and V. Madiseti. Bandwidth aggregation with SCTP. In *GLOBECOM'04*, volume 7, pages 3716–3721, 2004.
- [8] K.H. Kim, Y. Zhu, R. Sivakumar, and H.Y. Hsieh. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. *Wireless Networks*, 11(4):363–382, 2005.
- [9] D. Sarkar. A concurrent multipath TCP and its markov model. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 2, pages 615–620. IEEE, 2006.
- [10] J.R. Iyengar, P.D. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *Networking, IEEE/ACM Transactions on*, 14(5):951–964, 2006.
- [11] L. Magalhaes and R. Kravets. MMTP: multimedia multiplexing transport protocol. *ACM SIGCOMM Computer Communication Review*, 31(2 supplement):220–243, 2001.
- [12] F. Melakessou, U. Sorger, and Z. Suchanecki. Mptcp: Concept of a flow control protocol based on multiple paths for the next generation internet. In *Communications and Information Technologies, 2007. ISCIT'07. International Symposium on*, pages 568–573. IEEE, 2007.
- [13] K. Chebrolu, B. Raman, and R.R. Rao. A network layer approach to enable TCP over multiple interfaces. *Wireless Networks*, 11(5):637–650, 2005.
- [14] D.S. Phatak and T. Goff. A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 773–781. IEEE, 2002.
- [15] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. Mar: A commuter router infrastructure for the mobile internet. In *ACM MobiSys'04*, page 230.
- [16] K. Habak, M. Youssef, and K.A. Harras. DBAS: A Deployable Bandwidth Aggregation System. *Arxiv preprint arXiv:1201.2760*, 2012.
- [17] X. Zhang and K.G. Shin. E-mili: energy-minimizing idle listening in wireless networks. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 205–216. ACM, 2011.
- [18] B. Anand, K. Thirugnanam, J. Sebastian, P.G. Kannan, A.L. Ananda, M.C. Chan, and R.K. Balan. Adaptive display power management for mobile games. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 57–70. ACM, 2011.
- [19] M. Dong and L. Zhong. Chameleon: a color-adaptive web browser for mobile oled displays. *Arxiv preprint arXiv:1101.1240*, 2010.
- [20] J. Paek, K.H. Kim, J.P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using cell-id sequence matching. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 293–306. ACM, 2011.
- [21] M.B. Kjergaard, S. Bhattacharya, H. Blunck, and P. Nurmi. Energy-efficient trajectory tracking for mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 307–320. ACM, 2011.
- [22] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232. ACM, 2006.
- [23] W. Hua, J. Ohlund, and B. Butterklee. Unraveling the mysteries of writing a winsock 2 layered service provider. *Microsoft Systems Journal*, pages 96–113, 1999.
- [24] M. Carson and D. Santay. Nist net-a linux-based network emulation tool. *Computer Communication Review*, 33(3):111–126, 2003.
- [25] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *ACM WWW'07*.