Bandwidth Aggregation Techniques in Heterogeneous Multi-homed Devices: A Survey

Karim Habak, Georgia Institute of Technology
Khaled A. Harras, Carnegie Mellon University
Moustafa Youssef, Egypt-Japan University of Science and Technology (E-JUST) and Alexandria University

The widespread deployment of various networking technologies, coupled with the exponential increase in end-user data demand, have led to the proliferation of multi-homed, or multi-interface enabled, devices. These trends drove researchers to investigate a wide spectrum of solutions, at different layers of the protocol stack, that utilize available interfaces in such devices by aggregating their bandwidth. In this survey paper, we provide an overview and examine the evolution of bandwidth aggregation solutions over time. We begin by describing the bandwidth aggregation problem. We investigate the common features of proposed bandwidth aggregation systems and break them down into two major categories: layer-dependent and layer-independent features. Afterwards, we discuss the evolution trends in the literature and discuss some open challenges requiring further research. We end the survey with a brief presentation of related work in tangential research areas.

General Terms: Network Protocols, Internetworking, Network Architecture and Design, Wireless communication, Performance

Additional Key Words and Phrases: Computer networks, Bandwidth aggregation, Multi-homing, Multi-interface

1. INTRODUCTION

With the continuous advancement of wireless technologies, decreasing cost of electronics, increased audio and video Internet traffic [Gebert et al. 2012], and the heterogeneity of network access technologies, it is becoming the norm nowadays to have multi-interface enabled devices, also known as multi-homed devices. This provides an opportunity for leveraging these interfaces to meet the increased user demand on bandwidth. Unfortunately, state-of-art operating systems, such as Windows, Linux, MAC OS, and mobile devices operating systems, fail to utilize the true potentials of these interfaces. For instance, Windows and MAC OS are not equipped with any technique to utilize these interfaces concurrently. Linux, however, is equipped with true link equalizer (TEQL) which is designed enable using multiple interfaces to increase the bandwidth. Unfortunately, TEQL is not suitable for Internet communication and introduces many drawbacks that, in most cases, lead to system performance degradation. Therefore, Linux default environment disables TEQL and its users avoid using TEQL to leverage the available interfaces their multi-homed devices. On the other hand, Apple implemented a new multi-path-enabled transport layer protocol.
in iOS-7 and deployed it in millions of iPhones and iPads. This new protocol, however, leverages the available interfaces mainly when using Siri during its communication with Apple servers, otherwise, each interface is still independently used. Overall, the failure of state-of-art operating systems to leverage the available interfaces in multi-homed mobile devices, leads to the under-utilization of bandwidth, a waste of the potential connectivity resources, and user dissatisfaction.

The fundamental approach for leveraging multiple network interfaces on multi-homed mobile devices is aggregating the bandwidth available on each of these interfaces. We define bandwidth aggregation as the ability to leverage the available network interfaces to increase the bandwidth for users and applications. Over the past decade, a large body of research has emerged to enable bandwidth aggregation on multi-homed devices. To achieve their main goal of enhancing the user experience, they provide solutions for two major of challenges: (1) core aggregation challenges and (2) Internet integration challenges. First, core aggregation challenges refer to the set of challenges that come with bandwidth aggregation even when designed as a clean slate. Examples of these challenges include estimating network interface characteristics, and scheduling data across different interfaces. Second, Internet integration challenges refer to the set of challenges introduced by the current design of the Internet and its protocol layers. For instance, the vast majority of current Internet applications use TCP over IP as a result of historical decisions tightly linking both protocols. These decisions led to the development of systems and applications that ultimately expect to run on a single network interface, thus identifying communication end points by a single IP address at each end. As a result, many solutions have been developed and implemented at different layers of the TCP/IP protocol stack to solve the two challenges above and work around the current Internet design and characteristics.

In this paper, we survey the current state-of-the-art solutions for the bandwidth aggregation problem. We categorize, study, and share the various solutions implemented at different layers of the protocol stack. Solutions implemented in the same layer usually share common goals, challenges, and possess what we denote as layer-dependent features. On the other hand, there are other common features shared between all the solutions regardless of the layer they are implemented in. These layer-independent features include estimating the interface and applications characteristics, scheduling, and network support and communication model. In addition, we analyze the evolution of the solution space and discuss the open challenges and new trends. We note that while a previous survey for bandwidth aggregation has been conducted [Ramaboli et al. 2012], this survey however, lacks many key papers added in this survey. That survey also only focuses on scheduling and packet reordering challenges which is not sufficient for truly characterizing the efficiency of the bandwidth aggregation solutions. It also neglects deployment challenges, identifying different layer-dependent and independent features, shedding light on the chronological evolution of related literature over the past decade, and does not suggest enough open research challenges.

In addition to the core area surveyed in this paper, we also identify three tangential research areas that share some characteristics with bandwidth aggregation. First, multi-path routing addresses problems resulting from having multiple paths to a given destination [Tarique et al. 2009]. Second, resource aggregation in computer sub-systems, investigates obtaining higher performance by aggregating other computer resources such as hard disks [Katz et al. 1989]. Third, multiple network interfaces can be utilized for minimizing energy consumption [Shih et al. 2002], handling mobility [Bahl et al. 2004], controlling/redirecting traffic, using multiple channels [Draves et al. 2004], or avoiding primary users in cognitive radio networks [Habak et al. 2013].

The rest of the paper is organized as follows. Section 2 discusses the layer-dependent features. In Section 3, the layer-independent features are presented. Section 4 analyzes the area evolution over time as well as the open research challenges. In Section 5, we discuss other
tangential areas that have similar challenges and solutions. Finally, the paper concludes in Section 6.

2. LAYER-DEPENDENT FEATURES

A number of solutions have been proposed for the bandwidth aggregation problem on multi-homed devices. These solutions, shown in Figure 1, are implemented at different layers of the protocol stack which impacts the challenges they need to address and the features they can provide. In this section, we discuss the layer-dependent features of the different proposed techniques. We defer the discussion of the layer-independent features to the next section.

2.1. Application Layer Solutions

Apart from the multi-homed aware applications, which utilize available interfaces for their own benefit while lacking global efficiency and interface load awareness [Ananthanarayanan et al. 2007], application layer solutions are mainly implemented as a middleware between the application and transport layer. This middleware takes the responsibility of handling multiple interfaces and distributing the different application data across them. We break down these solutions into two main categories: (1) hidden middleware and (2) non-transparent middleware. Hidden middlewares have the advantage of ease of deployment and backward compatibility as they do not require changes to current applications. However, these solutions are usually more complex to implement compared to non-transparent middlewares. In addition, since applications are aware of the existence of the non-transparent middleware, the middleware can use more information from the applications to achieve better performance.

2.1.1. Hidden Middleware

Hidden middlewares are those who work seamlessly with current applications and enable them to make use of multiple available interfaces. In such cases, the middleware implements the same interface that the transport layer provides to the application. It also has to guarantee the same semantics provided by the traditional transport layer. For example, implementing the solution on top of a reliable transport layer protocol, such as TCP, must guarantee reliability. However, working on multiple interfaces may produce out-of-order delivery. Therefore, the solution needs to provide mechanisms for correct in-order packet delivery. To handle this, some solutions provide connection-level granularity scheduling where all the packets belonging to the same TCP connection will traverse the same interface while different connections can use multiple interfaces concurrently [Saeed et al. 2010; Habak et al. 2012b]. This approach enables them to operate with the conventional Internet architecture and servers. On the other hand, Sakakibara et al. [2006] and Miyazaki et al. [2012] implement a packet-reordering technique at both endpoints. Although these solutions show great potential in increasing system performance, updating the servers becomes a formidable barrier preventing their large-scale deployment.

To minimize the widespread deployment cost of the hidden middlewares, researchers avoid relying on updating the servers to enable bandwidth aggregation. Habak et al. [2012a; 2013; 2012c] use connection-level granularity scheduling as the default mode of operation while leveraging optional server modifications to further enhance performance. Moreover, Sharma et al. [2004; 2007] rely on a proxy server to hide the effects of using multiple interfaces at the client side form legacy servers. In this case, aggregating bandwidth occurs between clients and proxy servers. Furthermore, other researchers exploit application-layer protocol functionalities to aggregate the bandwidth of different interfaces without network or server support. For instance, Kaspar et al. [2010a] exploit the availability of HTTP range queries for bandwidth aggregation. In particular, They break an HTTP stream into several pieces and open a new TCP connection to get each piece separately, using an HTTP range query requesting each piece. To aggregate the interfaces bandwidth, they distribute these TCP connections across available network interfaces. Therefore, they are able to exploit these
Fig. 1: Bandwidth aggregation solutions and their corresponding location in the protocol stack.

On the other hand, motivated by the advent of dynamic adaptive streaming over HTTP (DASH) and its adoption by Youtube, Hulu, and Netflix, Evensen et al. [2011a; 2010] extend
the idea of using application-layer protocol functionalities for bandwidth aggregation by utilizing the DASH request-response communication model. In this case, instead of issuing different video-segment requests using the same interfaces, they distribute these requests across the different interfaces to enhance system performance and enable users to stream higher quality videos. Although, GreenBag [Bui et al. 2013] adopts the idea of distributing video requests across interfaces, they focus on minimizing the time of video playback along with energy consumption, while overlooking the ability to stream higher quality videos.

2.1.2. Non-transparent Middleware. Non-transparent middleware requires modifications to the applications in order to make use of available interfaces. In such cases, the middleware is able to modify some terms in the agreement between the application and the transport layer. These modifications aim to enhance the overall system performance as well as provide the application with what it needs. For example, MuniSocket [Mohamed et al. 2002] changes the agreement between the application and transport layer such that it supports reliability upon request by the applications. To do this, it uses UDP to transmit packets, and engages its implemented reliability mechanism if requested by the application. Intentional networking [Higgins et al. 2010] changes the agreement to guarantee the ordering of packets only upon a request from the application. The application defines its own ordering constraints and intentional networking guarantees satisfying these constraints. This guarantee is achieved by defining IROBs (Isolated Reliable Ordered ByteStreams) as the unit of data transmission defined by the application. Intentional networking gives each IROB a unique identifier. While creating an IROB, an application can specify a list of IROBs that have lower identifiers that must be received prior to receiving this IROB.

2.2. Transport Layer Solutions

As shown in Figure 1, many bandwidth aggregation solutions naturally lie in the transport layer. We classify these solutions into three categories: 1) extending widely deployed protocols (e.g. TCP), 2) utilizing multi-homing support in existing protocols (e.g. SCTP) and 3) designing new transport protocols.

2.2.1. Extending Widely Deployed Protocols. Since TCP has been the dominating protocol for Internet traffic for the past decades, a lot of work in the literature focuses on extending it to support transmission over multiple network interfaces. As a result, there exists many TCP extensions that support multiple interfaces. These protocols usually address issues that hinder TCP from utilizing multiple interfaces in parallel such as reordering packets belonging to different sub-flows, scheduling packets across multiple interfaces, node identification, and congestion control mechanisms.

Packet recording is usually performed using a global reordering buffer at the receiver. We defer the discussion about packet scheduling to the layer-independent features in Section 3.3. On the other hand, since a TCP connection is defined by the source and destination ports and IP addresses, all TCP extensions enable the source and/or the destination to have multiple IPs. These IPs are exchanged between the two endpoints in the beginning and during the lifetime of a connection. The rest of this section focuses on the TCP congestion control mechanisms because they are the core functionality of TCP and unique to the transport layer.

- Congestion Control: The key component of the TCP protocol is its congestion control mechanism that introduces most of its interesting characteristics such as fairness and ability to utilize the available bandwidth of the underlying network without overwhelming it. Therefore, several researchers develop congestion-control mechanisms to enhance TCP performance in different situations such as using wireless links [Sinha et al. 2002], utilizing satellite links [Henderson et al. 1997], and operating on top of high-bandwidth networks [Tan et al. 2006; Ha et al. 2008]. Such solutions are designed to utilize only one underlying
Fig. 2: Unfair bandwidth distribution between multipath TCP and regular TCP. Host 1 achieves double the bandwidth achieved by Host 2 because it uses multi-path variant of TCP, in which sub-flows control the congestion independent of each others.

path, and thus, take many design decisions such as 1) using out-of-order packet reception as an indicator for network-congestion that requires decreasing the transmission data rate and 2) using only one timeout variable to detect packet loss due to severe network congestion. Unfortunately, these design decisions are the main causes for the performance degradation of TCP while running on multiple heterogeneous interfaces. In this case, interface heterogeneity causes unnecessary timeouts and out-of-order packet delivery, leading to the unnecessary shrinking of the congestion window causing drop in transmission data rates. In addition, even if congestion was detected that affects an interface, the excessive decrease of the congestion window leads to decreasing the transmission rate on other interfaces as well. In many cases, running TCP on top of multiple interfaces achieves lower performance compared to running it on top of only one of these interfaces [Hsieh et al. 2002]. Therefore, solutions that extend TCP to support multi-interface communication modify its congestion control mechanism.

To efficiently utilize the available interfaces, TCP extensions modify the congestion control mechanism in different ways. For instance, many extensions propose applying a congestion control mechanism to each sub-flow independent of the other sub-flows, where a sub-flow is the portion of the connection that is sent over the same path. This technique implicitly decouples the congestion control problem from the packet reordering one. But when it comes to building their protocols, these extensions adopt different congestion control mechanisms based on the targeted usage scenario. For example, many protocols, e.g. [Barré et al. 2010; Diop et al. 2012; Hasegawa et al. 2005; 2007; Sarkar 2006; Paasch et al. 2012; Xue et al. 2010], deploy the standard congestion control mechanism of TCP on each sub-flow. In contrast, to achieve best utilization of high bandwidth links, Le et al. [2012b] use a cubic congestion control mechanism [Xu et al. 2004] and Le et al. [2012a] use a binomial congestion control mechanism [Bansal et al. 2001] for each of their sub-flows. Meanwhile, because of interface heterogeneity, pTCP [Hsieh et al. 2005] allows the use of a different TCP variant, e.g. [Henderson et al. 1997; Mascolo et al. 2001; Sinha et al. 2002], for each interface depending on its characteristics. Each of them runs its own congestion control mechanism independently and handles its interface characteristics accordingly. For example, using pTCP on a host that is equipped with a wireless WAN (WWAN) network interface and satellite interface will make it use WTCP [Sinha et al. 2002] for WWAN interface and STP [Henderson et al. 1997] for the satellite interface. Although these approaches enable TCP to efficiently utilize multiple interfaces in many scenarios, they lose its fairness property in case of having shared bottlenecks. Figure 2 shows a scenario, in which running a congestion control mechanism on each sub-flow independently leads to unfair distribution of the available bandwidth over running connections. In particular, this figure shows that multipath TCP [Barré et al. 2010] obtains double the bandwidth obtained by TCP.
To achieve fair bandwidth distribution at the shared bottlenecks while achieving the maximum utilization of the available interfaces, many researchers propose tweaks to the congestion control schemes. mTCP [Zhang et al. 2004] uses the standard congestion control mechanism of TCP on each sub-flow independent of the other sub-flows. However, when it detects that two or more sub-flows share their bottleneck, it merges them together as one sub-flow and uses one congestion window for them. Unfortunately, this approach is suitable only for long-lived connections since it takes seconds (maximum 15 seconds) to detect these shared bottlenecks. To overcome this problem, many protocols decouple the congestion control mechanisms, which controls the rates on the running sub-flows [Shamszaman et al. 2013; Wischik et al. 2011; Raiciu et al. 2012; Han et al. 2004; Honda et al. 2009; Peng et al. 2013]. Generally, these approaches limit the growth in the congestion window of each sub-flow based on various parameters such as the sum of the congestion windows of all sub-flows, the delay and congestion correlation with other sub-flows, and estimates of a competing-TCP throughput.

Instead of running different congestion control mechanisms at each sub-flow, cTCP [Dong et al. 2007] modifies TCP’s congestion control mechanism in order to deal with multiple interfaces. It uses a database at the sender to keep track of all the packets that have been sent but not acknowledged as well as the interface used for sending each packet. When an in-sequence packet acknowledgement is received, the sender deletes all the packets up to the one being acknowledged from the database. However, when a duplicate ACK is received with a packet number equal to \( n \), this highlights that the receiver is still waiting for the \((n+1)^{th}\) packet, while receiving subsequent packets to that. If the duplicate ACK is received over the same path used for sending the \((n+1)^{th}\) packet, this is considered a real duplicate acknowledgement due to packet loss. Otherwise, a race condition has occurred resulting in duplicate ACKs created due to differences in path characteristics. This intelligent handling of acknowledgments reduces the packet reordering problem.

2.2.2. Utilizing Multi-homing Support in Existing Protocols. SCTP [Stewart et al. 2000] is one of the protocols that researchers heavily investigated while proposing solutions for the bandwidth aggregation problem due to its inherent design that supports multi-streaming and multi-homing. Fortunately, SCTP allows data to be partitioned into multiple streams independently delivered to the application running at the receiver. This means that the loss of a data chunk belonging to a certain stream only affects the delivery within that stream. This feature prevents the head-of-line blocking problem that can occur in TCP, since TCP only supports single streams. Multi-homing also allows a single SCTP endpoint to support multiple IP addresses. SCTP multi-homing support, however, is only for redundancy. A single address is chosen as the primary address, which is the destination address for all data chunks during normal transmission. These characteristics of the SCTP protocol encouraged bandwidth aggregation researchers to work on extending it in order to exploit multiple interfaces in parallel.

Similar to extending TCP, work extending SCTP has focused on reordering packets belonging to different sub-flows or streams, scheduling packets across the different interfaces, and developing appropriate congestion control mechanisms. To discuss the detailed characteristics of these extensions, we categorize them into two main categories: (1) application-assisted aggregation, where applications provide some assistance to the SCTP extension such as defining their different streams or relations between their data units, and (2) application-oblivious aggregation, where the existence of multiple interfaces is hidden from the applications.

- Application-Assisted Aggregation: MCMT [Huang et al. 2010] is an extension to the SCTP protocol that uses an application-assisted aggregation approach. This extension utilizes the multi-streaming feature in the SCTP protocol to solve the previously mentioned challenges. It gives the applications the responsibility to define their streams. Then, it adopts
a path-oriented multi-streaming scheduling technique in which the packet that belongs to the same stream utilizes the same path. For example, an application streaming a video from a MCMT-enabled server is responsible for dividing its data into two streams. The first stream is used to transmit the video while the second is used for transmitting the related audio. In this case MCMT will transmit all video packets using the same path which may be different from the path used for transmitting the audio data. Assigning streams to paths is the task of the scheduler which is presented in Section 3.3. In addition, the application may further divide the video or audio into multiple streams to enhance performance. However, it will carry the overhead of reordering packets and applying its own reliability requirements on each stream.

- **Application-Oblivious Aggregation:** Many extensions to SCTP focus on developing an application-oblivious protocol that seamlessly aggregate the available bandwidth of the network interfaces. Therefore, they have to maintain the same contracts between applications and the SCTP protocol. Hence, they have to provide solutions to the previously mentioned problems. In particular, packet reordering is implemented by having a global reordering buffer for each stream. This buffer is used in case there are no requests for out-of-order packet delivery from the applications. In Section 3.3, we will address the scheduling mechanisms used in detail.

Because of the great similarities in adopted congestion control protocols between an SCTP stream and a TCP connection, SCTP extensions attempt to address the same challenges discussed in Section 2.2.1 while extending their congestion control mechanism. Similar to extending the congestion control of TCP, many solutions implement the standard SCTP stream congestion control mechanism at each sub-stream independent of the other sub-streams [Abd El Al et al. 2004b; 2004a; Argyriou et al. 2003; Liao et al. 2008; Casetti et al. 2004; Mirani et al. 2011; Mirani et al. 2010; Ye et al. 2004]. On the other hand, other SCTP extensions deploy a congestion control mechanism on the whole stream [Shailendra et al. 2012; Iyengar et al. 2006]. In this case, they changed the techniques of detecting congestion as well as updating the congestion window to be suitable for running over more than one interface. They change the fast retransmission technique such that a retransmission and a congestion windows decrease are triggered by out-of-order packet delivery for the packets that utilize the same path. Hence, they store the path used to send each packet, thus, when selective acknowledgement (SACK) identifies a gap in certain path, it triggers fast retransmission and congestion window update.

### 2.2.3. Designing New Protocols

While protocols presented in this section are newly designed, some of these protocols maintain the same application-transport contract of TCP due to its wide spread use and deployment.

Magalhaes et al. [2001c] propose the R-MTP transport layer protocol which uses retransmission-based reliability and gap-detection for identifying losses. The sender is notified that frames have arrived at the receiver by acknowledgements. R-MTP’s gap-detection relies on selective acknowledgment. In order to control the network congestion, R-MTP introduces a new congestion control mechanism: The receiver is the entity responsible for detecting congestion by monitoring the delay jitter calculated from the difference in time between every two consecutive packets and its mean value. This mean value is calculated from the rate which the sender and receiver agreed on. The idea is that, in the case of no congestion, the long term jitter should be close to zero. Hence, the increase in the delay jitter indicates network congestion. This congestion control technique is applied to each path independently.

RCP [Kim et al. 2005] is another example of newly designed protocols maintaining TCP’s contract between the application and the transport. RCP is a receiver-centric transport protocol designed to avoid the TCP limitations. It is implemented to deal with one network interface while keeping in mind the ease of extension to support multiple network interfaces.
The authors extend this protocol to support communication through multiple network interfaces by proposing R²CP. In RCP, reliability is implemented by making the receiver request data from the sender instead of acknowledging the data. They define two types of requests: 1) cumulative request, which is used in order to request new data, and 2) pull request, used to request packet retransmissions. Flow control in RCP is much easier than TCP where a receiver only sends requests when it has free space in its buffer. Congestion control is similar to TCP’s except for it being receiver-centric. R²CP applies this congestion control in each sub-flow to support additional interfaces.

Wang et al. [2009] divide the transport layer into two layers. In the top layer, they develop the multiple transmission control protocol (MTCP) responsible for awareness of the available paths, data stripping, data scheduling, and reliability. In the bottom layer, they use multiple common transport protocols (CTPs). Each CTP can be any transport layer protocol suitable for the network interface it will operate on. Furthermore, all the CTPs have to implement the same interface between it and the MTCP protocol. This interface enables MTCP to monitor the free spaces in their congestion window and act accordingly by pushing data to fill it.

MMTP [Magalhaes et al. 2001b] is designed to utilize the available interfaces in order to achieve the demanding multi-media’s bandwidth requirements. This protocol is designed in order to have the frame received before its deadline and avoid wasting network resources in sending frames that are going to be useless due to late arrival.

2.3. Network Layer Solutions

Network layer solutions target maintaining adopted and deployed transport layer protocols and allowing them to work efficiently on different network interfaces by making modifications to the network layer. Due to its popularity, network layer solutions usually use TCP as the target transport layer protocol. They consequently address three main issues that prevent TCP from achieving high performance while running on multiple interfaces: (1) breaking TCP’s connection semantics, (2) congestion misprediction, and (3) round trip time (RTT) estimation.

2.3.1. Breaking TCP Connection Semantics. Since any connection will have its packets distributed over multiple interfaces, and each interface has its own IP address, this breaks the TCP connection semantics that identifies a connection by the source and destination IPs and port numbers. To address this problem, network layer solutions hide the usage of multiple IPs from the running TCP protocol. For instance, Phatak et al. [2002] use IP-in-IP encapsulation to hide the usage of multiple IPs from TCP. In this case, the source and destination open a TCP connection with one IP for each of them. These IPs are used for all packets to/from the transport layer. When a packet is sent using another interface, or sent to an interface other than the one agreed on during connection establishment, the packet with the agreed upon IP from the transport layer is encapsulated in another packet whose header contains the actual interface IP. The network layer at the destination extracts this packet and forwards it to the destination transport layer. Fortunately, performance evaluation showed that the encapsulation overhead is negligible. On the other hand, to achieve the same goal, OSCAR [Habak et al. 2014] uses network address translation (NAT) instead of IP-in-IP encapsulation. In this case, instead of encapsulating packets with the negotiated IP addresses in other packets with the used IP addresses, OSCAR replaces the source and destination IPs at the sender with the used IPs for transmission. Upon receiving a packet, the receiver reverses the source and destination IPs by replacing them with the negotiated ones before giving the packet to TCP. Although the encapsulation and the NATing techniques show efficiency, implementing them requires updating the network layer at the endpoints.

To ease deployment, many solutions attempt to avoid updating the servers while proposing solutions that hide using multiple IPs from TCP. For example, Chebrouh et al. [2005]
rely on a proxy to hide client multiple IPs from the server. This proxy interacts with servers using a single IP address and is aware of the client’s multiple IPs while communicating with it. The solution adopts IP-in-IP encapsulation between the proxy and the client to hide the client IPs from the running TCP connection at the client side. On the other hand, Lan et al. [2012] adopt a different architecture which consists of a client, a server, a proxy server and a router equipped with multiple interfaces. They split a connection between the client and the server in three parts: (1) A normal TCP connection between the client and the router, (2) A normal TCP connection between the proxy server and the server, (3) multiple TCP connections between the router and the proxy server such that each of these connections utilizes only one path. In this case, they do not need to hide the used IPs from the running TCP connections. Furthermore, MAR [Rodriguez et al. 2004] uses a similar architecture with the following differences: (1) communication between the router and the proxy is not limited to using multiple TCP connections and (2) the proxy is optional to minimize the deployment cost. In the absence of a proxy, MAR provides a per-TCP connection mode of operation, in which each connection is assigned only to one interface but different connections can be assigned to different interfaces. MOTA [Deb et al. 2011], on the other hand, adopts a special case of connection-oriented scheduling in which all the application load is assigned to only one network, while different applications can be assigned to different networks.

We highlight that the problem of breaking TCP connections due to using multiple IP addresses appears in multiple contexts other than multi-interface bandwidth aggregation. For example, mobile devices change their IP addresses while moving, thus, handling user mobility and maintaining active connections also deal with this problem [Perkins 1997]. Therefore current state-of-the-art in these research areas can provide bandwidth aggregation researchers with mechanisms to solve this problem such as host identification protocol (HIP) [Nikander et al. 2010]. In addition, although some clean slate Internet architectures provide solutions to this problem [Han et al. 2012], they still introduce new set of challenges for bandwidth aggregation researchers.

2.3.2 Congestion Misprediction. When running TCP on top of multiple network interfaces that vary in terms of delay and available bandwidth, out-of-order packet delivery becomes the norm, leading to unnecessary drop in the congestion window and the transmission rate of TCP (Section 2.2.1). Therefore, hiding the out-of-order packet delivery from TCP is a critical feature of network layer bandwidth aggregation techniques. Some solutions solve this problem by implementing packet reordering within the network layer [Lan et al. 2012; Chebrolu et al. 2005; Manousakis et al. 2007; Manousakis et al. 2008; Habak et al. 2014]. Instead of delivering out-of-order packets to TCP, they buffer out-of-order packets in the network layer until the preceding packets arrive. Although this approach hides the out-of-order packet delivery from TCP and, thus, avoids the unnecessary shrinking of the congestion window, it should be carefully implemented since it may result in detecting packet loss only via timeout at the sender resulting in severe congestion window drops. Therefore, these solutions implement packet loss detection techniques at the network layer to avoid timing out on the lost packets. When a loss is detected at the network layer (e.g. by setting a threshold on the packet’s waiting time in the reordering queue), the network layer forwards the received out-of-order packets to TCP to trigger duplicate ACKs or selective ACKs (SACKs) which is used to detect the loss quickly and more importantly to avoid a timeout event, which is more costly than a duplicate ACKs or SACKs.

We note that MAR [Rodriguez et al. 2004] offers the architecture in which a protocol is used between a multi-homed router and the optional proxy. In order to achieve high performance, this protocol must be carefully designed such that it handles reordering and hides it from the end points. In addition, another approach to avoid this issue is using
connection oriented scheduling which is adopted by MAR in case of no proxy, and OSCAR while communicating with legacy servers.

2.3.3. Round trip time estimation technique. As a result of using multiple interfaces, each connection can go through multiple paths that vary in their behavior, including the round trip time. In addition, reordering affects the calculation of round-trip time (RTT) estimation and hence determining the right value for the retransmission timeout timer (RTO). Hence, Phatak et al. [2002] study the effect of distributing the data across the different network interfaces on the RTT and RTO estimation. They address this problem by building a mathematical model to avoid the negative effects of errors in estimating the RTT and determining the RTO and take this into account in their scheduling decision as we discuss in Section 3.3. Others handle this problem by implementing reordering at the network layer [Lan et al. 2012; Chebrolu et al. 2005; Manousakis et al. 2007; Manousakis et al. 2008; Habak et al. 2014]. This reordering delays the packets from the fast paths waiting for previous packets to arrive which were sent on the slow paths. This makes RTO and RTT estimations bound by the slowest path. On the other hand, connection-oriented scheduling provides another solution for this issue [Rodriguez et al. 2004; Habak et al. 2014].

2.4. MAC Layer Solutions
MAC layer solutions are the first bandwidth aggregation solutions to emerge to address problems such as providing enough communication bandwidth between database servers. There are special cases of bandwidth aggregation solutions that work between two devices directly connected through multiple links. MUP [Adya et al. 2004] is one of the MAC layer protocols designed to aggregate the bandwidth of multiple radios tuned to different channels while communicating with a neighbor. In this protocol, scheduling works on spectrum availability bases such that it sends the frame directly on the available free spectrum. Furthermore, other approaches developed scheduling techniques that can efficiently utilize the identical available interfaces independent of the transmission media [Malkin 1999; Adya et al. 2004]. SRR tried to relax the identical assumption of the network interfaces by assuming that they use the same media while having different capacities [Hari et al. 1999]. On the other hand, Koudouridis et al. [2005] introduced a generic link layer (GLL) approach to use multiple radios while communicating with a certain destination. Their approach is unique because it considered using an intermediate relay node while communicating with the destination. They assume that they have two kinds of radios: (1) radios that are directly connected to the destination, and (2) radios that are connected to the destination through a one-hop relaying node. They distribute the data across these radios based on spectrum availability and capacity. On the other hand, Lin et al. [2004] aggregate multiple Ethernet links by introducing a bonding layer, below the network layer, responsible for distributing packets across these Ethernet links. In order to distribute packets across Interfaces, they assign a packet to an interface once the interface finishes transmitting its old packet and becomes ready to send a new packet. They also extend the ARP protocol to implement ARP+ which enables maintaining multiple MAC addresses for the same IP address.

3. LAYER-INDEPENDENT FEATURES
We define the layer-independent features as the set of features shared by bandwidth aggregation solutions regardless of the layer in which they are implemented. These features include: (1) interfaces characteristic estimation, (2) applications characteristics estimation, (3) scheduling, and (4) network support and communication model.

Before we start addressing each feature separately, we notice that there exists a group of solutions that have huge design similarities. Therefore we grouped them in Table I. The main reason behind these similarities is that they use their different congestion control mechanisms for scheduling packets across different network interfaces and aim at utilizing
the interfaces' available bandwidth to its maximum. We call these solutions congestion-based protocols (CBP).

3.1. Interface Characteristics Estimation

Interface characteristics estimation is considered one of the most important features of any bandwidth aggregation system. It is responsible for capturing the heterogeneity of the different network interface, including traffic load, loss rate, interface capacity, etc. In this section, we discuss the interface characteristics estimation techniques used by various solutions as well as the different approaches proposed to estimate each of them.

3.1.1. Bandwidth estimation. Estimating the available bandwidth at each interface is a key functionality for bandwidth aggregation systems since it is the most popular metric taken into account when scheduling data across different interfaces. The most dominant techniques proposed for such estimation are: (1) delay jitter based estimation, (2) packet pair, (3) interface traffic monitoring, (4) active and passive probing, (5) operator-assisted estimation, and (6) implicit estimation.

- Delay-Jitter-Based Estimation: Magalhaes et al. [2001c] implement a delay jitter based bandwidth estimation technique that is the receiver’s responsibility. This technique is based on an agreement regarding the transmission data rate between the sender and receiver. The receiver estimates the delay jitter based on the inter-arrival time between packets (Figure 3). The average long term jitter should hover around zero since it takes...
Fig. 3: Negative and positive inter-arrival time jitter values caused by the variance in propagation delay while average jitter hovers around zero [Magalhaes et al. 2001c].

positive and negative values as shown in Figure 3. This jitter will increase in the case of congestion. Based on this increase, the receiver detects congestion and notices that the available bandwidth is less than the utilized data rate. The receiver also estimates the reception data rate (available bandwidth) and sends it to the sender. However, using this technique, the sender can only reduce its sending rate but cannot detect the increase of the available bandwidth. This leads to a waste of the available bandwidth unless detected using other bandwidth estimation techniques. Overcoming this problem can be done by having a protocol periodically probe the paths using other techniques such as the packet-pair method [Keshav 1995].

- **Packet-Pair:** The packet-pair technique [Keshav 1995] is one of the main bandwidth estimation techniques used by several bandwidth aggregation systems [Habak et al. 2014; Chebrolu et al. 2005; Evensen et al. 2009; Tachibana et al. 2012; Sakakibara et al. 2006]. In addition, R-MTP [Magalhaes et al. 2001c] uses it to overcome some of the shortcomings mentioned above. In this technique, the sender sends two back-to-back packets on each path. These packets are served by the path bottleneck, which leads to spacing them out in time. Once a packet arrives at the destination, an ACK is directly sent back to the source. These ACKs will preserve the same time spacing between the reception of the packets. By measuring the inter-arrival time between the ACKs, the available bandwidth at the path bottleneck can be estimated. The sending rate can then be adjusted based on this estimate.

Note that to gather accurate bandwidth estimates using the packet-pair technique, the sender should use long packet trains (not only two packets) and measure the average inter-arrival time between every two consecutive ACKs. To avoid the overhead of probing the network with long packet trains, the systems that adopts packet-pair techniques use application data packets to probe the network.

- **Interface Traffic Monitoring:** Several approaches rely on monitoring the different network interfaces to estimate the bandwidth [Saeed et al. 2010; Habak et al. 2012c; 2012a; 2012b; 2013]. They estimate the available bandwidth by measuring the average number of bytes sent and received per interface when running TCP connections. This is based on the fact that TCP congestion control enables it to transmit at a data rate close to the available bandwidth. Although this technique accurately estimate the bandwidth of a downlink, It is not suitable for estimating the bandwidth of an uplink in case of having UDP streams uploading traffic concurrently with TCP connections.
Active and Passive Probing: Assuming that the bottleneck is local, Habak et al. [2012a; 2013; 2012b; 2012c] use active probing techniques to estimate the available bandwidth at each interface. They periodically connect to various geometrically dispersed servers in order to estimate the uplink and the downlink available bandwidth. They also combine these estimates with statistics collected during the normal data transfer by interface traffic monitoring. When running in packet-oriented mode, where each packet can be scheduled to a different interface, DBAS [Habak et al. 2012a; 2013] and OPERETTA [Habak et al. 2012c], obtain a better estimate of the available bandwidth by probing the actual destination.

Intentional Networking [Higgins et al. 2010] uses a similar method where bandwidth estimation is based on randomly probing selected geographically spanned reference servers [Nicholson et al. 2006]. They also introduce a passive probing mode, where data packets, if available, are used to probe the network [Kim et al. 2001]. This mode monitors the exchanged data packets between a source and destination, and estimates the available bandwidth accordingly between them using exponential averaging. Furthermore, they deploy four filters on their observations in order to quickly detect network status changes while resisting transients in these observations.

Operator-assisted estimation: For bandwidth estimation, MOTA [Deb et al. 2011] makes the network operator aid the host in estimating the needed bandwidth from this operator. The operator broadcasts information about its available bandwidth and current load. This information is then used by the host to estimate the bandwidth it will utilize if its load traverses the corresponding interface. This approach is based on the willingness of network operators to share accurate information about their available bandwidth and current load.

Implicit estimation: Although congestion-based protocols (CBP) adopt different congestion control mechanism (Section 2.2), they depend on these mechanisms to keep the transmission rate at each interface close to its available bandwidth. In this case they do not need to explicitly estimate the available bandwidth for each interface. Unfortunately, this technique is only applicable when developing a reliable, congestion-aware, and multi-interface-aware transport layer protocol which would face a steep deployment barrier.

3.1.3. Delay. End-to-end delay is estimated by bandwidth aggregation solutions in order to be used in loss detection and/or scheduling. Congestion-based protocols (CBP) and systems, such as [Mohamed et al. 2002; Lan et al. 2012; Hasegawa et al. 2005; Evensen et al. 2009; Mirani et al. 2011; Mirani et al. 2010], estimate the round trip time (RTT) as the average time difference between sending a packet and receiving its acknowledgement and use it to calculate the retransmission timeout (RTO). On the other hand, to take the delay into account while distributing traffic across interfaces, Phatak et al. [2002] argue that it is sufficient to calculate the differences in latency between the interfaces, not the actual latency of each interface. To avoid requiring time synchronization that is infeasible, all time calculations are measured at the sender. The sender sends multiple packets at the same time on different interfaces and calculates the difference between receiving their ACKs. To increase the accuracy of this calculation, they force the receiver to send all the ACKs using a single path. Hence, these ACKs will face the same delays and the dominant part of their reception time difference is due to the difference of the main packets reception. ATLB [Hasegawa et al. 2005] also estimates the sender’s queuing delay as the ratio between the queue length of each sub-flow along with its average achieved throughput. Cross-layer FPS [Mirani et al. 2011] uses a cross layering approach to estimate the MAC layer contention incurred delays (ex: backoff time).

3.1.3. Energy Consumption. Some systems aim to combine maximizing the overall system throughput with minimizing the overall energy consumption. Hence, estimating energy consumption rates of the each network interface becomes one of their critical tasks. Habak et
al. [2012b; 2012c; 2014] rely on the fact that energy consumption is based only on the interface’s NIC. Hence, they save the various energy consumption rates of different network cards in a database. Interface characteristics estimation modules can then query this database in order to estimate the interfaces’ energy consumption rates. On the other hand, GreenBag [Bui et al. 2013] models the energy consumption of wireless interfaces (Wifi and LTE) as a function of the transmission time and the bandwidth used, as well as other constant factors that are technology dependent. Their model takes into account the energy consumed in the active transmission/reception states and the TAIL state.

3.2. Applications Characteristics Estimation

A number of solutions use their knowledge about the applications characteristics to enhance scheduling decisions. These solutions can be classified into three main categories: (1) qualitative input, (2) quantitative input, and (3) estimation.

3.2.1. Qualitative input. In this case, the system takes hints from the applications to enhance their scheduling technique. For example, Intentional Networking [Higgins et al. 2010] asks the applications to determine their type (foreground or background) and their transmission load (small or large) from some predefined categories. Others take into consideration the application’s required transmission type (reliable or not) [Abd El Al et al. 2004b; 2004a; Argyriou et al. 2003; Liao et al. 2008; Casetti et al. 2004; Mirani et al. 2010; Mirani et al. 2011; Ye et al. 2004].

3.2.2. Quantitative input. Other systems ask the application to explicitly define its required bandwidth [Magalhaes et al. 2001b; Zhu et al. 2004]. On the other hand, MOTA [Deb et al. 2011] takes application weights from the user to know the importance of the application. Bandwidth is assigned to the applications according to their relative weight.

3.2.3. Estimation. Other more deployable solutions estimate the applications requirements instead of explicitly asking the applications to determine them. This has the advantage of being backwards compatible and transparent to the applications. For example, Saeed et al. [2010] and Habak et al. [2012a; 2013; 2012c] estimate the connection’s sent and received bytes based on the application connection’s history. As a result, the estimated connection length equals the average connection length calculated from the history. Another technique used by these systems is based on the application name/type. For example, Skype is treated as a realtime application while an FTP client is treated as a bulk transfer application. Because application layer protocols generally have reserved ports for their communication, OSCAR [Habak et al. 2014] uses a similar approach but instead of maintaining these estimates for each application, it maintains it for each port. On the other hand, GreenBag [Bui et al. 2013] monitors the status of the video player and estimates whether it is playing or buffering. It also estimates the video quality, played time, and remaining time.

3.3. Scheduling

Scheduling data across different interfaces is one of the core tasks in any bandwidth aggregation system. In this section, we address two aspects of scheduling: scheduling granularity and scheduling techniques.

3.3.1. Scheduling granularity. Scheduling granularity refers to the unit of data that can be assigned to a network interface. There are two categories for scheduling granularity: packet-level and connection-level.

- Packet-Level Scheduling: To achieve optimality, most bandwidth aggregation solutions adopt packet-level scheduling granularity, in which packets belonging to the same connection can be assigned to different interfaces. This requires support from both ends of
the connection, or the introduction of proxy servers, and usually leads to higher throughput performance.

- **Connection-Level Scheduling:** To ease the deployment, some bandwidth aggregation solutions adopt connection-level scheduling granularity, in which different connections can be assigned to different network interfaces. Packets belonging to the same connection must be assigned to the same network interface. These solutions either utilize it as the main operational mode [Saeed et al. 2010; Habak et al. 2012b] or as an optional mode triggered by the lack of network support [Rodriguez et al. 2004; Habak et al. 2012a; 2013; 2012c]. MOTA [Deb et al. 2011] is considered a special case of connection-oriented scheduling in which connections belonging to the same application are assigned to the same network interface. The main advantage of connection-oriented scheduling is backwards compatibility with legacy servers. In contrast, Habak et al. [2012a] show that connection-level scheduling granularity can significantly enhance performance if and only if connection lengths are taken into account while scheduling. Otherwise, it can lose its advantage, and in some cases lead to performance degradation.

### 3.3.2. Scheduling techniques

In this section we present the most prominent scheduling techniques that have been proposed to distribute data across different interfaces.

- **Round Robin:** Round-robin scheduling is used in MAR [Rodriguez et al. 2004], and adopted as a baseline technique for comparison in a number of systems, e.g. [Saeed et al. 2010; Habak et al. 2012a; 2012c]. In this technique, data is assigned to interfaces in a rotating basis without taking into account the capacity of the interface or the application requirements. SRR [Hari et al. 1999] investigated a queue-size-based variant of the round-robin scheduler. In this variant, the schedule iterates on the interfaces in a rotating basis assigning each packet to the interface that has free slots in their queue.

  Habak et al. [2012a; 2013; 2012b; 2012c] also investigate weighted round robin scheduling in both connection-oriented or packet-oriented scheduling modes. LS-SCTP [Abd El Al et al. 2004b; 2004a] also uses weighted round robin scheduling where weights are the ratio between the congestion window size and the round trip time. Others use another variant of weighted round robin in which weights are determined by the available bandwidth of each path [Evensen et al. 2009; Nguyen et al. 2013; Kim et al. 2008].

  Another solution is based on a mathematical model to determine the fraction of packets that should be sent by each interface without degrading TCP performance [Phatak et al. 2002]. The idea is to make all interfaces have the same timeout value. This is based on scheduling packets over different interfaces based on their relative bandwidth, similar to the weighted round robin technique. However, contrary to weighted round robin, which has a fixed packet size, the proposed solution has a different packet size for each interface to guarantee the same timeout value on all interfaces.

- **Maximum Throughput:** Assuming we are bound by connection-level granularity scheduling, some systems [Habak et al. 2012a; 2013] introduce a maximum throughput scheduling technique. This technique aims to maximize the overall system bandwidth without considerations to the bandwidth of a specific connection or stream. It works in the connection-oriented granularity mode. For a new connection, the scheduler assigns it to the network interface that will maximize system throughput. This is equivalent to assigning the new connection to the interface that minimizes the time needed to finish the current system load in addition to the load introduced by this new connection. This algorithm depends on the estimated connection length and the estimated bandwidth for each interface.

  If packet-level granularity scheduling is possible, minimizing the packet delivery time reflects the increase in overall system throughput. This approach maximizes the stream/connection throughput while minimizing the reordering overhead. Packet-pair based earliest-delivery-path-first scheduling sends packets in pairs on the path, which will deliver
Bandwidth Aggregation Techniques in Heterogeneous Multi-homed Devices: A Survey

Fig. 4: SBAM data distribution relative to the interfaces bandwidth-delay product (BDP) [Sakakibara et al. 2006].

it in the shortest time to the destination [Chebrolu et al. 2005]. Westwood SCTP (W-SCTP)[Casetti et al. 2004] estimates the chunk’s delivery time at each network interface in order to select the interface that has the shortest delivery time to serve that chunk. This procedure is done for each chunk until the congestion windows of the available paths have been exhausted. FPS [Mirani et al. 2010] dynamically estimates the packet delivery time at each network interface and fills this difference with in-order data to avoid packet reordering. Cross-layer FPS [Mirani et al. 2011] extends FPS [Mirani et al. 2010] using a cross-layering approach to include the MAC layer contention delays (ex. backoff delay) while estimating the packet delivery time at each interface. ATLB [Hasegawa et al. 2005; 2007] ranks the interfaces such that the minimum score interface has the minimum delivery time, and assigns packets to the interface with the minimum score. ATLB calculates the score of each interface using the following equation:

\[
score_i = \frac{Q_i}{G_i} + \frac{RTT_i}{2}
\]

such that \(Q_i\) is the queue length of sub-flow \(i\), \(G_i\) is the average throughput of sub-flow \(i\) and \(RTT_i\) is its average round trip time.

- **Bandwidth Delay Product:** SBAM [Sakakibara et al. 2006] schedules data based on the bandwidth delay product (BDP) of each network interface. The technique starts by sending data on the interface with the maximum BDP. If the other end supports SBAM, the system leverages the other interfaces and packets are distributed over the different interfaces according to their BDP.

- **Leveraging Congestion Control:** Distributing packets across different interfaces can also be done based on the congestion control mechanism. Although congestion-based protocols (CBP) adopt different mechanisms (Section 2.2), they share the concept of using their congestion control mechanism for scheduling. In this case they apply congestion control mechanisms on each interface, or on the whole connection while dividing the congestion window over the interfaces and assigning packets to interfaces only when they have empty space in their congestion window. This technique maintains the transmission rate on each interface close to its full capacity to increase the overall system throughput. Furthermore, E2EMPT [Xue et al. 2010] uses a path priority assignment scheme in order to be used while assigning packets to interfaces in case of having multiple interfaces with free slots in their congestion windows.

- **Rate-based:** Magalhaes et al. [2001c; 2001a] use rate-based techniques to schedule packets. After estimating the available bandwidth for all network interfaces, this technique calculates the packet rate that each interface can support. It sends packets on each interface
with the rate supported by this interface. Such technique relies on accurately estimating the capacity of each interface.

- **Energy and Cost Aware:** G-DBAS [Habak et al. 2012b] proposes two types of energy-aware scheduling techniques: an energy efficient scheduler and a utility-based scheduler. The energy-efficient scheduler assigns connections to the interface that minimizes the overall energy consumption. The utility-based scheduler, however, combines the maximum throughput and energy efficient schedulers using a user defined utility function to achieve different user goals.

  OPERETTA [Habak et al. 2012c] combines energy efficiency with throughput maximization by formulating the scheduling problem as a linear program. Its main target is to minimize the energy consumption while achieving a certain amount of throughput. This throughput is calculated from a user defined utility function that indicates the user’s willingness to increase energy consumption for more throughput. OPERETTA also combines the packet-oriented mode with the connection-oriented mode reaching the optimal target without updating any all destination servers.

  GreenBag [Bui et al. 2013] schedules the real-time video streaming traffic across interfaces to support the required quality of service in the most energy efficient way. It divides the video file to chunks and downloads each chunk on the interface that minimizes both playback time and energy consumption.

  OSCAR [Habak et al. 2014] combines energy efficiency and cost efficiency with throughput maximization by proposing multi-objective and multi-modal scheduling. Based on the operational mode, OSCAR tries to optimize one parameter while maintaining the other parameters in the user acceptable ranges. it also combines the packet-oriented mode with the connection-oriented mode reaching the optimal target without updating all destination servers.

- **Quality of Service:** Another approach is application-required QoS based packet scheduling. For instance, in QAWBA, when aiming for collaborative bandwidth aggregation
between peers [Zhu et al. 2004], each node is assumed to have one interface connected to the Internet and another connected with its peers. The application first defines its required bandwidth. Then the scheduler reserves as much as possible from its local link and sends requests to other nodes to reserve the extra bandwidth needed along with the maximum number of hops for this request. Scheduling packets is performed to fit the reserved bandwidth at each path. Figure 5 shows an example of QAWBA where five mobile nodes form a MANET. The client node $C$ runs an application requiring 500Kbps, of which only 300Kbps are available from its cellular link. $A$ and $D$ act as proxies to forward a portion of the total traffic to $C$. The 500Kbps traffic flow is split into three flows at the base station, and then forwarded to $C$ via different paths. Thus, with the help of nodes $A$ and $D$, $C$ is able to receive the required 500Kbps bandwidth by aggregating three flows, which would not be possible under one single cellular connection.

MMTP [Magalhaes et al. 2001b] provides another scheduling technique designed for multimedia applications that have hard deadlines in delivering frames. This technique selects the best interface to send a frame based on its estimated arrival time using this interface and its arrival time deadline. For a given frame, it searches for network interfaces that have non-utilized available bandwidth. This is achieved in the protocol by making each interface generate tokens according to its own available bandwidth. The tokens are used based on the following rules:

— If no token is available, the frame must wait.
— If exactly one token is available, and the estimated arrival time is before the frame arrival deadline, then send the frame using this interface. Otherwise, wait until either a new token appears to check for its interface, or the deadline is reached after which it will be dropped.
— If more than one token is available, select the interface which has the highest propagation delay that can deliver the packet on time. This keeps the interface channel filled and increases the probability that a new packet will be delivered on time.

3.4. Network Support and Communication Model

Different systems adopt different client-server communication models, so they require different levels of network support. These models are selected to achieve certain system objectives such as increasing the overall system performance and minimizing the system adoption cost. The proposed systems generally adopt three communication models that we visually summarize in Figure 6: 1) updated-server-based communication, 2) proxy-based communication and 3) legacy-server-based communication.

3.4.1. Updated-Server-based Communication Model. This communication model mainly targets increasing the overall system performance without requiring any updates to the core network infrastructure. Therefore, it relies on implementing bandwidth aggregation solutions at the communication end-points (the client and the server). Hence, many bandwidth aggregation systems adopt this model either in their normal operation mode [Sklover et al. 1996; Higgins et al. 2010; Hari et al. 1999; Le et al. 2012a] or as an optional mode for improving the system’s performance [Habak et al. 2012a; 2012c; 2013]. With the flexibility of updating communicating end points in this model, the ability to design fine-grained schedulers and accurate interface characteristics estimators increases. On the other hand, this approach carries the overhead of updating destination servers in order for it to be adopted.

3.4.2. Proxy-based Communication Model. Another communication model is to avoid or minimize server updates by placing a proxy server that is aware of the client’s multiple interfaces and aids the client in its bandwidth aggregation. In such cases, the proxy is assumed to be connected to destination servers via high bandwidth links. Most solutions adopting this communication model mainly use it to avoid updating end servers [Tachibana et al. 2012;
Sharma et al. 2004]. PRISM [Kim et al. 2005] deploys this proxy server in order to minimize the updates in the communication server.

MAR’s optional mode of operation is a special case of adopting this model [Rodriguez et al. 2004]. The main difference is that they do not assume that clients are equipped with multiple network interfaces, but instead, are connected to a MAR router equipped with multiple network interface. The MAR router works with the proxy server in this optional mode in order to utilize its available network interfaces and enhance the overall system performance.

Overall, this model of communication enables clients to use fine-grained scheduling techniques as well as efficient interface characteristics estimation solutions. On the other hand, a proxy introduces another set of challenges such as where to place it and how to avoid multiple clients contending for the proxy server, which can ultimately render the proxy server itself becoming a bottleneck.

3.4.3. Legacy-Server-based Communication Model. Driven by the need to avoid infrastructure and server updates, some work has adopted this communication model. Although this model lacks the ability to deploy fine-grained scheduling techniques, its coarse-grained (connection-oriented) scheduling enables clients to utilize available interfaces and achieve high performance gains. Researchers adopted this communication model while developing bandwidth aggregation systems that put the responsibility of exploiting interfaces on the client devices which has the minimum updating cost [Habak et al. 2012c; Deb et al. 2011].
4. EVOLUTION AND CHALLENGES

After providing an overview of bandwidth aggregation systems and their features, this section discusses the evolution of these systems along with current challenges that remain to be addressed by the research community.

4.1. Evolution

Bandwidth aggregation solutions have evolved over the past decade in different forms. We chronologically present the most prominent research conducted in this area in Table II, and compare them based on the set of parameters shown in the table. In addition, we discuss in this section two forms of evolution: Protocol stack layered evolution and scheduling granularity evolution.

4.1.1. Protocol Stack Layered Evolution. In the beginning, the need for increasing available bandwidth coupled with the ability to be equipped with multiple network interfaces was only the case in data centers. Researchers proposed increasing the bandwidth of database servers in these centers by connecting them with multiple identical wired cables. Hence, they started implementing bandwidth aggregation solutions at the MAC layer in order to make use of these multiple links [Sklower et al. 1996].

With the exponential growth in technology and decreasing cost of electronics, the number of multi-homed devices exponentially increased. These devices include normal desktops, laptops, tablets and smart phones. Researchers started developing solutions for utilizing the available interfaces on such devices. Hence, this problem started to look like an end-to-end network problem rather than a single-hop problem. This is why MAC layer solutions could not be adopted in such cases.

Researchers began to define this problem as a transport layer problem that requires the modification or replacement of current transport layer protocols with multipath aware protocols. They started by proposing new multi-path transport layer protocols [Magalhaes et al. 2001b] and utilizing the multihoming support in existing transport layer protocols like SCTP [Argyriou et al. 2003]. Although many solutions have been proposed, well-known single path transport protocols (e.g. TCP) were already heavily adopted and deployed. Therefore, researchers started proposing modifications to TCP to utilize the available interfaces [Hsieh et al. 2005]. Because of the formidable deployment barrier these approaches faced, researchers had to find a work-around to overcome this problem through network or application layer approaches.

Therefore, researchers developed network layer solutions that can utilize the network interfaces while hiding them from transport layer protocols to avoid performance degradation [Phatak et al. 2002]. Others developed application layer solutions that utilize multiple transport protocol sessions in order to exploit available interfaces [Mohamed et al. 2002]. The main drawback of such solutions is the need for updating either the legacy servers or network infrastructures. This drawback encouraged researchers to focus on developing application layer solutions on stand alone devices. Therefore more recent application layer solutions that do not require updating end-servers or network infrastructure appeared [Habak et al. 2012a].

4.1.2. Scheduling Granularity Evolution. Once researchers started to think about providing bandwidth aggregation solutions, they were aiming for the optimal performance. Hence, packet level scheduling was initially adopted. All solutions adopted this fine-grained level of scheduling in order to maximize performance gains. This level of scheduling, however, introduced the challenge of high deployment cost as a result of the need for updating end-servers, network infrastructure, as well as client applications. This fact is largely the reason we do not see any pervasive bandwidth aggregation solutions to date.
| System                  | Sched. Gran. | Network Stack Layer | Deployability | App. Ch. Knowledge |
|-------------------------|--------------|---------------------|---------------|--------------------|
|                         | Packet Level | Con. Level | App. Layer | Tran. Layer | Net. Layer | MAC Layer | Client Upd. | Server Upd. | Infra. Sup. | Input | Est |
| MP [Sklower et al. 1996]| ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| MMP [Malkin 1999]      | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| SFRK [Hari et al. 1999]| ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| MMTP [Magalhaes et al. 2001b] | ✓              | ✓       | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| R-MTP [Magalhaes et al. 2001c] | ✓              | ✓       | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Phatak et al. [2002]   | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| MuniSocket [Mohamed et al. 2002] | ✓            | ✓       | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| SCTP [Argyriou et al. 2003] | ✓     | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Bondingplus [Lin et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| MCP [Adya et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| MAR [Rodriguez et al. 2004] | ✓              | ✓       | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| QAWBA [Zhu et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Abd’ElAl et al. [2004a; 2004b] | ✓              | ✓       | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| IPCC-SCTP [Ye et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| W-SCTP [Casetti et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| mTCP [Zhang et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Handhielded R. [Sharina et al. 2004] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| GLI [Koudouridis et al. 2005] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| ‘Chebrotu et al. [2005] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| yTCP [Hsieh et al. 2005] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| R2CP [Kim et al. 2005] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| ATLB [Hasegawa et al. 2005] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| PRISM [Kim et al. 2005] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| CMT [Iyengar et al. 2006] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| cmpTCP [Sarkar 2006]    | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| SBAM [Sakakibara et al. 2006] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Manousakis et al. [2007] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| WiMP-SCTP [Huang et al. 2007] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| cTCP [Dong et al. 2007] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| ATLB [Hasegawa et al. 2007] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Handhielded R. [Sharina et al. 2007] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Ananthanarayanan et al. [2007] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| Manousakis et al. [2008] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| cmpRTCP [Anand et al. 2008] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |
| cmpSCTP [Liao et al. 2008] | ✓            | ✓         | ✓          | ✓           | ✓          | ✓         | ✓           | ✓           | ✓          | ✓     | ✓   |

Table II: Evolution (✓ means required and ✡ means optional)
| System | Sched. Gran. | Network Stack Layer | Deployability | App. Ch. Knowledge |
|---------|--------------|---------------------|--------------|--------------------|
|         | Packet Level | Con. Level | App. Layer | Tran. Layer | Net. Layer | MAC Layer | Client Upd. | Server Upd. | Infra. Sup. | Input | Est |
| Evensen et al. [2009] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Wang et al. [2009] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MPTCP [Barré et al. 2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| E2EMPT [Xue et al. 2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MCMT [Huang et al. 2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| FFS [Mirani et al. 2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Intent. Net. [Higgins et al. 2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Kaspar et al. [2010a] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Kaspar et al. [2010b] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Evensen et al. [2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DNIS [Saeed et al. 2010] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Evensen et al. [2011b] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MOTA [Deb et al. 2011] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Wischik et al. [2011] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| X-Layer FPS [Mirani et al. 2011] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MPSCTP [Shaileendra et al. 2011] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Evensen et al. [2011a] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| ETOM [Lan et al. 2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Miyazaki et al. [2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MPSCTP [Kalci et al. 2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Diop et al. [2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Kin et al. [2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MPCubic [Le et al. 2012b] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Le et al. [2012a] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Cao et al. [2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Miyazaki et al. [2012] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| G-DBAS [Habak et al. 2012b] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DBAS [Habak et al. 2012a] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| OPERETTA [Habak et al. 2012c] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Nguyen et al. [2013] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MPTCP [Feng et al. 2013] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| MPTCP [Shamszaman et al. 2013] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| GreenBag [Bui et al. 2013] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DBAS [Habak et al. 2013] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| OSCAR [Habak et al. 2014] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table II: Evolution (√ means required and ✠ means optional) (Continue)
Researchers had to think about this problem differently. Rodriguez et al. [2004] noticed that applications tend to have many connections that can be scheduled across available interfaces at connection-level granularity to enhance throughput without modifying end-servers. Their main drawback is that they did not design a new connection-based scheduling technique, but used simple round-robin and weighted round-robin techniques which do not guarantee performance enhancement in the long run.

At this stage, some researchers focused on minimizing the cost of using packet level scheduling techniques, while others developed more novel connection-level scheduling techniques [Saeed et al. 2010; Habak et al. 2012a; 2012c]. For example, OPERETTA [Habak et al. 2012c] combines both connection level scheduling with packet level scheduling to achieve the maximum performance without updating all legacy servers.

4.2. Challenges

In this section, we discuss some open research challenges which we believe can make use of further investigation by researchers in the community.

4.2.1. Deployability. Deploying a bandwidth aggregation system without modification to current infrastructure and devices is one of the most important challenges that prevent most proposed solutions from achieving their ultimate goal. This challenge was not addressed during the past decade until recent work has taken the first steps in doing so [Saeed et al. 2010; Habak et al. 2012a; Hasegawa et al. 2007]. In this section we address the deployability obstacles that face current solutions and how some solutions have addressed these obstacles. There are four main barriers hindering deployability at different levels: 1) intermediate device usage, 2) client updates, 3) server updates, and 4) application updates. After addressing these deployment barriers we shed the light on some attempts to deploy bandwidth aggregation systems.

- Intermediate Device Usage Using intermediate devices in order to implement bandwidth aggregation solutions limits the adoption of such solution. Requiring such devices, e.g. routers and proxy servers, increases deployment cost. In addition, with the widespread adoption of such solutions, these devices would be performance bottlenecks at edge networks and would need to scale accordingly.

MAR [Rodriguez et al. 2004] uses a router as well as an optional proxy to implement their solution. The router is the device to which their multiple interface clients are attached. Clients connect to this router and set it as their DNS server. The optional proxy’s existence changes MAR’s mode of operation from connection level scheduling to packet level scheduling.

MOTA [Deb et al. 2011] requires updating the network operator’s base stations in order to support devices with the information required to schedule the data across different interfaces. It requires the operator to support these devices with information about its current load, the cumulative weight of all the applications assigned to it, and the available bandwidth. Hence, the device uses this information to estimate its available bandwidth.

Other solutions, such as [Hasegawa et al. 2007; Chebrolu et al. 2005], rely on the existence of proxy servers to implement their techniques. In these solutions, client devices connect to the proxy, with the scheduling technique running between them.

On the other hand, other solutions (e.g. [Diop et al. 2012; Sakakibara et al. 2006]) avoid using intermediate devices and implement their solutions only on communicating end-points to minimize their deployment cost. These solution are more deployable in this sense.

- Server Updates: Updating the server restricts the widespread deployment of a solution. It is difficult to update legacy servers around the world in order to make use of multiple interfaces at the client. Updating these servers requires large amounts of money, effort, and time.
Some solutions lost deployability by requiring server updates [Iyengar et al. 2006; Barré et al. 2010]. These solutions focus on building an end-to-end system updating both clients and servers to utilize the available interfaces to their maximum. To increase deployability, other solutions avoid updating the servers [Saeed et al. 2010; Habak et al. 2012b]. Habak et al. [2012c; 2014], however, have two modes of operation. Connection-oriented mode in which they do not require any updates to the servers in order to increase deployability and a packet-oriented mode which makes use of optional server updates in order to maximize the overall system performance.

- Application Updates: Updating legacy applications greatly impacts bandwidth aggregation system deployment. Generally, bandwidth aggregation solutions implemented below the transport layer do not interface with applications [Phatak et al. 2002; Chebrolu et al. 2005]. Hence, they are backwards-compatible with current applications. On the other hand, transport layer solutions and application layer solutions differ from one another based on the degree of backward compatibility they offer.

Transport layer protocols that change the contract or interface between the applications and the transport layer are generally not backwards compatible and require updating legacy applications [Magalhaes et al. 2001b]. Other solutions aim at replacing currently used protocols with new multi-interface aware protocols or extending current protocols to add this awareness [Hsieh et al. 2005; Hasegawa et al. 2005]. Although such solutions maintain the same interface and contract terms between the applications and transport layer, they require updating the applications since the only feasible way of deploying them in the current operating system is as a new transport protocol that lies beside the old set of protocols to maintain backwards-compatibility with Internet servers. cTCP [Dong et al. 2007] solves this issue by implementing a TCP extension that is backwards-compatible with the current TCP by maintaining its interface with the application. Such extension can be deployed in the current operating system as a replacement of the TCP protocol while avoiding the need for updating applications.

On the other hand, different approaches for implementing application-layer solutions exist (Section 2.1). Hidden-middleware based solutions do not require updating the applications since they maintain the same interfaces between the applications and their transport layer protocols [Saeed et al. 2010; Sakakibara et al. 2006]. Non-transparent-middleware based solutions, however, generally require updating the applications because such middleware either updates the interface between the applications and transport layer or requires certain input from these applications [Mohamed et al. 2002; Higgins et al. 2010].

- Client Updates: Updating clients also impacts deployability. This effect is minimal, however, compared to the other updates above, since it is normal to update devices with new patches nowadays. The impact of this factor depends on the complexity of the updates themselves.

Updating the kernel is considered the most complex and expensive update on the client side. Some solutions require updating the client kernel by modifying the network protocol stack [Barré et al. 2010; Kim et al. 2005]. These updates decrease the deployability of the solutions. Others required installing some software at the user level without the need for recompiling the kernel [Habak et al. 2012a; Saeed et al. 2010]. This installation adds a new layer, which takes the responsibility of utilizing the available interfaces. Rodriguez et al. [2004] and Hasegawa et al. [2007] avoid updating client devices by implementing their solution in the router. It is only required to configure the client device by setting the default DNS server to be the router itself.

- Deployment Attempts: Throughout the last decades, there were many attempts to deploy bandwidth aggregation systems in different contexts. For instance, the emergence of data centers has increased the need for high bandwidth communication between datacenter
nodes. The lack of technology support for such bandwidth led to the emergence and the deployment of link bonding mechanisms and MAC layer bandwidth aggregation solutions at these data centers. These approaches were adopted because the nodes in the data centers are either directly connected using multiple Ethernet cables or connected through a set of switches. After the successful deployment of link bonding mechanisms in data centers, Linux implemented true or trivial link equalizer (TEQL) [Li et al. 2006], a link bonding technique to enable users to utilize their multiple interfaces. Unfortunately, TEQL is only suitable for directly connecting devices with multiple homogeneous interfaces as well as connecting a device to a gateway with multiple homogeneous links. Therefore, the nature of having multiple heterogeneous interfaces at mobile devices led to performance degradation while using TEQL. Hence, TEQL is not enabled by default in the current versions of Linux.

Over time, distributed data centers and cloud services emerged. This trend opened a new set of communication challenges such as the need for more bandwidth and the need for certain reliability levels. Fortunately, multi-path communication provided a solution for such challenges, thus, new transport layer protocols that utilize the available paths to achieve desired data-center goals emerged. For instance, these transport layer protocols use some paths to enhance reliability through replicating the sent packets and use others to increase the communication bandwidth. The nature of having control on both communication ends at these data-centers led to the successful adoption of multi-path transport layer protocols. However, these data centers did not follow specific standards while implementing their transport layer protocols. Therefore, each data-center developed its multi-path transport protocol that achieves its desired goals.

With the availability of multi-homed mobile devices, their nature of having heterogeneous interfaces, and the increasing user demand for bandwidth, two attempts of deploying bandwidth aggregation systems are currently in action: (1) Apple implemented a variant of multi-path TCP in iOS 7 and (2) IETF is in final stages of having a standard for multi-path TCP. Although Apple successfully deployed their multi-path TCP protocol [Ford et al. 2013] in millions of mobile devices in the first few weeks, it is not considered a successful deployment yet since it works only with Apple servers while using Siri application. On the other hand, IETF developed a new multi-path TCP standard [Handley et al. 2013] and released it asking for comments and feedbacks. Although Apple demonstrated the ability of updating millions of devices in few weeks, the two deployment attempts relies on the willingness of Internet server operators to deploy these versions of multi-path TCP at their side.
4.2.2. Utilizing Middle-boxes. Although relying on middle-boxes such as proxy servers increases the deployment cost, the existence of content distribution networks demonstrates the acceptance of this cost if it results in tremendous performance enhancement. To date, researchers focus on using middle-boxes to enable bandwidth aggregation and address many challenges such as scheduling and interface characteristics estimation. However, they overlook many challenges introduced by using middle-boxes and potential usage of these middle-boxes such as 1) scalability and fault tolerance, 2) proxy placement, 3) traffic redirection, and 4) performance optimization.

- **Scalability and Fault Tolerance:** With the widespread deployment and adoption of middle-box based solutions, the available bandwidth and the computation power at these middle-boxes would become the performance bottleneck leading to performance degradation at the client side. Hence, to handle many devices, researchers should give sufficient attention to the scalability of their approaches and their design. In addition, efficiently handling middle-box failures is critical to avoid decreasing the quality of user experience.

- **Proxy Placement:** Since the client Internet traffic will pass through the middle-boxes, the location of these middle-boxes can significantly impact performance. Optimally, these middle-boxes should be placed on the route between clients and servers and should be connected to the servers via high speed links. Such placement avoids significant increase in client-server latency but it is almost infeasible because of the widespread deployment of Internet servers with whom a single client communicates at any point in time. Therefore, we argue that placing middle-boxes at the edge of Internet close to clients minimizes the latency overhead. In contrast, placing these middle-boxes at the Internet edge increase the deployment cost due to the need of more middle-boxes to cover all the Internet edge networks. Hence, researchers should handle the performance-cost tradeoff introduced by their middle-box placement techniques.

- **Traffic Redirection:** Seamless traffic redirection mechanisms is a critical component for distributing the load across the different middle-boxes and handling faults. Designing an efficient traffic redirection mechanism is considered a tremendous challenge because it should consider multiple aspects such as 1) maintaining end-clients and end-servers communication status, 2) maintaining the client interface information, and 3) avoid performance degradation while redirecting the traffic.

- **Performance Optimization:** Using middle-boxes enables many performance optimization techniques that significantly enhance the performance. For instance, caching the server contents can significantly enhance the response time. On the other hand, using middle-boxes enables adopting many communication protocols that are suitable for the client and can avoid performance degradation due to interface characteristics (e.g. loss in wireless environments). This enables data bundling and compression at the client or the middle-box to make best use of the client’s limited bandwidth. Furthermore, It enables utilizing opportunistic networks like Wifi in bus stations.

4.2.3. Enabling Client Collaboration. Although most bandwidth aggregation work focuses on utilizing the interfaces on a single device, there are a few attempts that target exploiting the bandwidth available on interfaces in neighboring devices [Sharma et al. 2007; Zhu et al. 2004; Ananthanarayanan et al. 2007; Krishnaswamy et al. 2012; Habak et al. 2014] (Figure 7). The motivation behind this collaborative approach is the increased mobile device density, the availability of high speed ad-hoc links amongst them and the lack of continuous, reliable, high bandwidth, and cheap Internet connectivity. In such cases, each device has at least two different network interfaces. The first interface is directly connected to an expensive limited bandwidth to Internet while the other interface can be used in collaboration between the
different devices. This collaboration introduces a new set of challenges including 1) neighbor discovery, 2) user incentives, 3) security issues, 4) data scheduling, and 5) sharing caches.

- **Neighbor Discovery**: One of the most important features of a collaborative bandwidth aggregation system is the ability of a device to discover its neighbors and gather essential information about them (e.g. their ability to share bandwidth and how much they are willing to share). Available solutions investigate the neighbor discovery problem in different ways. QAWBA [Zhu et al. 2004] implements a k-hop neighbor discovery protocol in which a QAWBA node discovers an Internet sharing node within k-hops. Other approaches discover neighboring devices using a proxy server that handles device collaboration [Sharma et al. 2007; 2004; Kim et al. 2005].

Unfortunately, developed approaches to date discover neighbors and return their shared bandwidth assuming that this bandwidth comes from a single interface. This assumption is not always true since the neighboring device may be equipped with multiple interfaces connected to the Internet and already sharing these different connections that have different characteristics. Recently, OSCAR [Habak et al. 2014] took initial steps towards solving this problem by enabling devices to return their different connectivity options that they are willing to share. Although OSCAR enables devices to share bandwidth that is shared with them from other devices, in many cases, OSCAR nodes fail in discovering shared bandwidths that are reachable through multiple hops. On the other hand, using a proxy to discover neighboring devices and handle collaboration is not efficient since it introduces communication overhead and synchronization problems between clients and the proxy. It also limits the widespread deployment of the system since proxies can be a bottleneck in this case.

- **User Incentives**: It is critical for a collaborative bandwidth aggregation system to be equipped with an incentive system to encourage users to share their bandwidth. Creating and integrating effective incentive systems with bandwidth aggregation solutions can only lead to the popularity and the widespread deployment of the system. Borrowing techniques and ideas from other incentive system solutions is a good way to start. These systems can be categorized into three categories: (1) game-theoretic based systems; (2) reputation-based systems; and (3) credit-based systems.

  Game-theoretic based incentive systems rely on the rationality of the game players. These approaches design a game in which the collaborating nodes will not gain or even lose if they try to cheat or do not collaborate [Jaramillo et al. 2010; Yao et al. 2011]. They assume that all the nodes have global knowledge about the game status and can interact accordingly. Such assumption creates an obstacle for adopting this type of incentive schemes in the context of large scale collaborative bandwidth aggregation systems as maintaining and spreading this information will introduce a huge overhead.

  In reputation-based systems, each node builds a reputation by serving other nodes in order to be served in the future. In such systems, each node carries the overhead of monitoring its neighbors since they most probably will collaborate with them later. It is also responsible for spreading gathered observation regarding neighbors through the network to enable the other nodes determine their reputation levels and act accordingly. Sharma et al. [2007] used this approach to provide incentives for their small collaborative community. This approach does not scale, however, and prevents such systems from being fully deployed over the Internet.

  On the other hand, credit-based systems are suitable for large scale networks since they usually rely on a trusted third party that maintains credits for the communicating nodes [Zhu et al. 2009; Chen et al. 2010]. These nodes usually collaborate with each other and every node pays for the service it requests. The collaborating nodes who offer the service usually gain credit which they use for their own benefit. OSCAR [Habak et al. 2014] uses
this approach to provide incentive for its users. This advantage makes this kind of incentive system best suitable for widely used collaborative bandwidth aggregation systems.

- **Security**: Security is one of the most important challenges in a collaborating environment. Selfish node behavior may drive them towards cheating in order to exploit the collaborating nodes connectivity without sharing their own resources. More importantly, nodes may eavesdrop, alter, or maliciously compromise relayed data. Connections, if not properly authenticated can be hijacked and users may be impersonated. These kinds of challenges should be efficiently addressed and security-based solutions need to be adopted and tailored for bandwidth aggregation environments.

- **Scheduling**: Scheduling on neighboring devices interfaces adds a lot of challenges including monitoring their gain as well as implementing deployable and seamless relaying. Schedulers need to address other metrics like incentive cost, energy consumption across devices, and fair throughput for all devices. Schedulers will also need to decide on which interfaces to use for scanning and sharing connectivity.

- **Sharing Caches**: To enhance performance and achieve better utilization of the available bandwidth of Internet connections, collaborating devices can share their caches together. For instance, sharing DNS caches is an approach to increase the responsiveness of Internet applications. On the other hand, sharing HTML-5 caches may significantly enhance the performance of HTML-5 applications and online games. It avoids wasting the bandwidth of Internet connections in transferring redundant data.

5. **TANGENTIAL RESEARCH AREAS**

There are three tangential research areas close to the multi-interface bandwidth aggregation problem as shown in (Figure 8): (1) multi-path routing, (2) resources aggregation in computer sub-systems, and (3) utilizing the availability of multiple network interfaces for non-bandwidth aggregation purposes.

5.1. **Multi-path Routing**

Multi-path routing refers to using multiple routes between the source and the destination [Tarique et al. 2009]. The multiple routes are selected to go through multiple neighbors, which is usually performed for reliability and security reasons. Data can be replicated on the different paths [Wang et al. 2005], distributed on multiple paths [Liang et al. 2005; Popa et al. 2006], encoded to enhance the reliability [Yang et al. 2010], or sent using a single path, while maintaining other paths as backup [Lim et al. 2003].

Multi-path QoS routing inherits all the characteristics from the multi-path routing while making use of the available multiple paths to grant a certain QoS level to the different streams [Wang et al. 2007; An et al. 2005]. These approaches use multiple metrics to quantify the QoS given to the streams such as delay and bandwidth.

5.2. **Resource Aggregation in Computer Sub-systems**

Aggregating available resources to obtain higher performance is a known problem in traditional computer systems [Katz et al. 1989; Traw et al. 1995]. Data stripping is the key aspect in the Redundant Arrays of Inexpensive Disks (RAID) architecture [Katz et al. 1989] for disk sub-systems. Traw et al. [1995] provide an overview on resource aggregation in network sub-systems and introduce an evaluation criteria to measure the benefits of the aggregated resources in terms of latency, buffering requirements, skew tolerance, scalability, complexity, and finally the maximum aggregate bandwidth which can be supported. These network sub-systems handle communication between different processors, processor and memory, as well as different hosts. They take the TCP/IP protocol stack network sub-system as a case study. However, they did not address the implementation layer, scheduling, nor evaluation.
5.3. Utilizing Multiple Network Interfaces

Utilizing the available multiple network interfaces has been an active research area during the last several years. In this survey we address utilizing them for bandwidth aggregation. However, there exists a large body of research work that utilizes these interfaces to achieve other goals.

Some researchers utilize the available interfaces in order to minimize using the highly loaded cellular networks. The Wiffler system [Balasubramanian et al. 2010] opportunistically offloads data over WiFi to minimize the use of these cellular networks when they become heavily loaded. Wiffler was developed since the currently used techniques aimed to encourage users minimize cellular network load, such as imposing a limit of 5 GB per month or educating users on responsible network access, are deemed ineffective and insufficient.

Other researchers exploit multiple interfaces in order to minimize energy consumption. Some of them use the interface with low energy consumption to wake up other interfaces [Shih et al. 2002; Banerjee et al. 2007; Bahl et al. 2004]. Johansson et al. [2002] leverage these interfaces to reduce the energy consumption as well. They show that Bluetooth radios are often preferable to IEEE 802.11 for short-range communication.

Others utilize these interfaces to handle mobility and overcome the wireless challenges [Bahl et al. 2004]. They utilize identical wireless interfaces in order to increase the seamless-ness of the wireless access point handoff. They propose techniques that tolerate the wireless link problems and avoid drawbacks of random backoff. They propose using one interface to control the media and prepare the schedule which other interfaces follow to transmit its data. They also propose increasing the communication capacity by using multiple wireless interfaces tuned to different channels.

Finally, some researchers utilize the available interfaces in the context of cognitive radio and mesh networks. In cognitive radio networks, they leverage these interfaces to avoid
interfering with primacy users and to create better spectrum opportunities [Brik et al. 2005; Ma et al. 2005; Karmoose et al. 2013]. In addition, some interfaces can be dedicated and used for control traffic [Habak et al. 2013]. In the context of mesh networks, Draves et al. [2004] show how the overall throughput can be increased for multi-radio nodes by dynamically choosing the “best” outbound link when forwarding a given packet.

6. CONCLUSION
In this paper we have surveyed the most prominent solutions proposed for addressing bandwidth aggregation problems in multi-homed devices. We have discussed the problem, examined and analysed the proposed research, and showed its tangential areas. We analyzed the different features of the problem solution and discussed how each solution implemented this feature. Finally, we have analyzed the various evolution trends and discussed potential open challenges we believe researchers can pay attention to.

REFERENCES
A. Abd El Al, T. Saadawi, and M. Lee. 2004a. Improving throughput and reliability in mobile wireless networks via transport layer bandwidth aggregation. *Computer Networks* (2004).
A. Abd El Al, T. Saadawi, and M. Lee. 2004b. L5-SCPT: a bandwidth aggregation technique for stream control transmission protocol. *Computer Communications* (2004).
A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. 2004. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *IEEE BroadNets*. IEEE.
H.Y. An, X-C. Lu, Z. Gong, and W. Peng. 2005. A cluster-based QoS multipath routing protocol for large-scale MANET. *High Performance Computing and Communications* (2005).
J. Anand and D. Sarkar. 2008. Architecture, Implementation, and Evaluation of a Concurrent Multi-path Real-time Transport Control Protocol. In *IEEE MILCOM*. IEEE.
G. Ananthanarayanan, V.N. Padmanabhan, C.A. Thekkath, and L. Ravindranath. 2007. Collaborative downloading for multi-homed wireless devices. In *Mobile Computing Systems and Applications, 2007. HotMobile 2007. Eighth IEEE Workshop on*. IEEE.
A. Argyriou and V. Madisetti. 2003. Bandwidth aggregation with SCTP. In *IEEE GLOBECOM*. IEEE.
P. Bahl, A. Adya, J. Padhye, and A. Walman. 2004. Reconsidering wireless systems with multiple radios. *ACM SIGCOMM Computer Communication Review* (2004).
A. Balasubramanian, R. Mahajan, and A. Venkataramani. 2010. Augmenting mobile 3G using Wifi. In *ACM MobiSys*. ACM.
N. Banerjee, M.D. Corner, and B.N. Levine. 2007. An energy-efficient architecture for DTN throwboxes. In *IEEE INFOCOM*. IEEE.
D. Bansal and H. Balakrishnan. 2001. Binomial congestion control algorithms. In *IEEE INFOCOM*. IEEE.
S. Barré, O. Bonaventure, C. Raiciu, and M. Handley. 2010. Experimenting with multipath TCP. In *ACM SIGCOMM Computer Communication Review*. ACM.
V. Brik, E. Rozner, S. Banerjee, and P. Bahl. 2005. DSAP: a protocol for coordinated spectrum access. In *IEEE DySPAN*. IEEE.
D. H. Bui, K. Lee, S. Oh, I. Shin, H. Shin, H. Woo, and D. Ban. 2013. GreenBag: Energy-efficient Bandwidth Aggregation for Real-time Streaming in Heterogeneous Mobile Wireless Networks. In *IEEE RTSS*. IEEE.
Y. Cao, M. Xu, and X. Fu. 2012. Delay-based congestion control for multipath TCP. In *IEEE ICNP*. IEEE.
C. Casas and W. Gaiotto. 2004. Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling. In *IEEE VTC Fall*. Ieee.
K. Chebrolu, B. Raman, and R.R. Rao. 2005. A network layer approach to enable TCP over multiple interfaces. *Wireless Networks* (2005).
B. B. Chen and M. C. Chan. 2010. MobiCent: a Credit-Based Incentive System for Disruption Tolerant Network. In *IEEE INFOCOM*. IEEE.
S. Deb, K. Nagaraj, and V. Srinivasan. 2011. MOTA: engineering an operator agnostic mobile service. In *ACM MobiCom*. ACM.
C. Diop, G. Dugué, C. Chassot, and E. Exposito. 2012. QoS-oriented MPTCP extensions for multimedia multi-homed systems. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE.
Y. Dong, D. Wang, N. Pissinou, and J. Wang. 2007. Multi-path load balancing in transport layer. In Next Generation Internet Networks, 3rd EuroNGI Conference on. IEEE.

R. Draves, J. Padhye, and B. Zill. 2004. Routing in multi-radio, multi-hop wireless mesh networks. In ACM MobiCom. ACM.

K. Evensen, D. Kaspar, P. Engelstad, A.F. Hansen, C. Griwodz, and P. Halvorsen. 2009. A network-layer proxy for bandwidth aggregation and reduction of IP packet reordering. In Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on. IEEE.

K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A.F. Hansen, and P. Engelstad. 2011a. Using bandwidth aggregation to improve the performance of quality-adaptive streaming. Signal Processing: Image Communication (2011).

K. Evensen, D. Kaspar, A.F. Hansen, C. Griwodz, and P. Halvorsen. 2011b. Using multiple links to increase the performance of bandwidth-intensive UDP-based applications. In Computers and Communications (ISCC), 2011 IEEE Symposium on. IEEE.

K. Evensen, T. Kupka, D. Kaspar, P. Halvorsen, and C. Griwodz. 2010. Quality-adaptive scheduling for live streaming over multiple access networks. In Proceedings of the 20th international workshop on Network and operating systems support for digital audio and video. ACM.

A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (2013).

S. Gebert, R. Pries, D. Schlosser, and K. Heck. 2012. Internet Access Traffic Measurement and Analysis. In Traffic Monitoring and Analysis. Springer.

S. Ha, I. Rhee, and L. Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review (2008).

K. Habak, M. Abdelatif, H. Hagras, K. Rizc, and M. Youssef. 2013. A Location-Aided Routing Protocol for Cognitive Radio Networks. In IEEE ICNC. IEEE.

K. Habak, K. A. Harras, and M. Youssef. 2014. OSCAR: A Collaborative Bandwidth Aggregation System. ArXiv e-prints (Jan. 2014).

K. Habak, M. Youssef, and K.A. Harras. 2012a. DBAS: A Deployable Bandwidth Aggregation System. IFIP NTMS (2012).

K. Habak, M. Youssef, and K.A. Harras. 2012b. G-DBAS: A Green and Deployable Bandwidth Aggregation System. In IEEE WCNC. IEEE.

K. Habak, M. Youssef, and K.A. Harras. 2012c. OPERETTA: An Optimal Energy Efficient Bandwidth Aggregation System. In IEEE SECON. IEEE.

K. Habak, M. Youssef, and K.A. Harras. 2013. An optimal deployable bandwidth aggregation system. Computer Networks (2013).

D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G Andersen, and others. 2012. XIA: Efficient support for evolvable internetworking. USENIX NSDI (2012).

H. Han, S. Shakkottai, C.V. Hollot, R. Srikant, and D. Towsley. 2004. Overlay TCP for multi-path routing and congestion control. In IMA Workshop on Measurements and Modeling of the Internet.

M. Handley, O. Bonaventure, C. Raiciu, and A. Ford, 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (2013).

A. Hari, G. Varghese, and G. Parulkar. 1999. An architecture for packet-stripping protocols. ACM Transactions on Computer Systems (TOCS) (1999).

Y. Hasegawa, I. Yamauchi, T. Hama, H. Shimomishi, and T. Murase. 2005. Improved data distribution for multipath TCP communication. In IEEE GLOBECOM. IEEE.

Y. Hasegawa, I. Yamauchi, T. Hama, H. Shimomishi, and T. Murase. 2007. Deployable multipath communication scheme with sufficient performance data distribution method. Computer Communications (2007).

T Henderson and R Katz. 1997. Satellite transport protocol (STP): An SSCOP-based transport protocol for datagram satellite networks. In Proceedings of Workshop on Satellite-Based Information Services. Citeseer.

B.D. Higgins, A. Reda, T. Alperovich, J. Flinn, TJ Giuli, B. Noble, and D. Watson. 2010. Intentional networking: opportunistic exploitation of mobile network diversity. In ACM MobiCom. ACM.

M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda. 2009. Multipath congestion control for shared bottleneck. In Proc. PFLDNeT workshop.

H.Y. Hsieh and R. Sivakumar. 2002. pTCP: An end-to-end transport layer protocol for striped connections. In IEEE ICNP. IEEE.
Bandwidth Aggregation Techniques in Heterogeneous Multi-homed Devices: A Survey

H.Y. Hsieh and R. Sivakumar. 2005. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks* (2005).

C.M. Huang, M.S. Lin, and L.H. Chang. 2010. The Design of Mobile Concurrent Multipath Transfer in Multihomed Wireless Mobile Networks. *Comput. J.* (2010).

C.M. Huang and C.H. Tsai. 2007. WiMP-SCTP: Multi-Path Transmission Using Stream Control Transmission Protocol (SCTP) in Wireless Networks. In *Advanced Information Networking and Applications Workshops, 2007,AINAW’07.* IEEE.

J.R. Iyengar, P.D. Amer, and R. Stewart. 2006. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *Networking, IEEE/ACM Transactions on* (2006).

J. J. Jaramillo and R. Srikant. 2010. A game theory based reputation mechanism to incentivize cooperation in wireless ad hoc networks. *Ad Hoc Networks* (2010).

P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla. 2002. Personal area networks: Bluetooth or IEEE 802.11? *International Journal of Wireless Information Networks* (2002).

M. Karmoose, K. Habak, M. El-Nainay, and M. Youssf. 2013. Dead Zone Penetration Protocol for Cognitive Radio Networks. In *IEEE WMob.* IEEE.

D. Kaspar, K. Evensen, P. Engelstad, and A.F. Hansen. 2010a. Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces. In *IEEE ICC.* IEEE.

D. Kaspar, K. Evensen, P. Engelstad, A.F. Hansen, P. Halvorsen, and C. Griwodz. 2010b. Enhancing video-on-demand playout over multiple heterogeneous access networks. In *IEEE CCNC.* IEEE.

R.H. Katz, G.A. Gibson, and D.A. Patterson. 1989. *Disk system architectures for high performance computing.* University of California, Berkeley, Computer Science Division.

S. Keshav. 1995. A control-theoretic approach to flow control. *ACM SIGCOMM Computer Communication Review* (1995).

H.A. Kim, B. Oh, and J. Lee. 2012. Improvement of MPTCP Performance in heterogeneous network using packet scheduling mechanism. In *IEEE APCC.* IEEE.

K.H. Kim and K.G. Shin. 2005. Improving TCP performance over wireless networks with collaborative multi-homed mobile hosts. In *ACM MobiSys.* ACM.

K.H. Kim, Y. Zhu, R. Sivakumar, and H.Y. Hsieh. 2005. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. *Wireless Networks* (2005).

M. Kim and B. Noble. 2001. Mobile network estimation. In *ACM MobiCom.* ACM.

P.S. Kim, J.Y. Yoon, and H. Kim. 2008. A packet partition scheduling mechanism for bandwidth aggregation over multiple paths. *Journal of Convergence Information Technology* (2008).

G.P. Koudouridis, R. Agüero, E. Alexandri, J. Choque, K. Dimou, HR Karimi, H. Lederer, J. Sachs, and R. Sigle. 2005. Generic link layer functionality for multi-radio access networks. In *IST Mobile and Wireless Communications Summit.*

D. Krishnaswamy, D. Zhang, D. Cavendish, W. Ge, S.S. Soliman, B. Mohanty, and S. Eravelli. 2012. COBA: Concurrent Bandwidth Aggregation-A Case Study in Parallel Wireless Communications. *Journal of Communications* (2012).

K.C. Lan and C.Y. Li. 2012. Improving TCP performance over an on-board multi-homed network. In *IEEE WCNC.* IEEE.

T.A. Le, C.S. Hong, and LEE Sungwon. 2012a. Multipath Binomial Congestion Control Algorithms. *JEICE Transactions on Communications* (2012).

T.A. Le, HAW Rim, C.S. Hong, and LEE Sungwon. 2012b. A Multipath Cubic TCP Congestion Control with Multipath Fast Recovery over High Bandwidth-Delay Product Networks. *JEICE Transactions on Communications* (2012).

Ji Li and Jack Brassil. 2006. Performance characterization of traffic equalizers on heterogeneous communication links. *Submitted to Networking* (2006).

Y. Liang and S.F. Midkiff. 2005. Multipath fresnel zone routing for wireless ad hoc networks. In *IEEE WCNC.* IEEE.

J. Liao, J. Wang, and X. Zhu. 2008. cmpSCTP: An extension of SCTP to support concurrent multi-path transfer. In *IEEE ICC.* IEEE.

H. Lim, K. Xu, and M. Gerla. 2003. TCP performance over multipath routing in mobile ad hoc networks. In *IEEE ICC.* Ieee.

Hsin-hung Lin, Chih-wen Hsueh, and Guo-Chiuan Huang. 2004. Bondingplus: real-time message channel in Linux Ethernet environment using regular switching hub. In *Real-Time and Embedded Computing Systems and Applications.* Springer.
L. Ma, X. Han, and C.C. Shen. 2005. Dynamic open spectrum sharing MAC protocol for wireless ad hoc networks. In *IEEE DySPAN*. IEEE.

L. Magalhaes and R. Kravets. 2001a. End-to-end inverse multiplexing for mobile hosts. *Journal of the Brazilian Computer Society* (2001).

L. Magalhaes and R. Kravets. 2001b. MMTP: multimedia multiplexing transport protocol. *ACM SIGCOMM Computer Communication Review* (2001).

L. Magalhaes and R. Kravets. 2001c. Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts. *Urbana* (2001).

G. Malkin. 1999. *Nortel networks multi-link multi-node PPP bundle discovery protocol*. Technical Report. September 1999. RFC 2701.

K. Manousakis and D. Famolari. 2008. INTELiCON: A Framework for the Simultaneous Utilization of Multiple Interfaces and its Application on TCP. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC’08. International*. IEEE.

K. Manousakis, P. Gopakakrishnan, D. Famolari, and E. Van Den Berg. 2007. INTELiCON: Intelligent Connectivity Framework for the Simultaneous Use of Multiple Interfaces. In *IEEE ICC*. IEEE.

S. Mascolo, C. Casetti, M. Gerla, M.Y. Sanadidi, and R. Wang. 2001. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In *ACM MobiCom*. ACM.

F.H. Mirani, N. Boukhatem, and M.A. Tran. 2010. A data-scheduling mechanism for multi-homed mobile terminals with disparate link latencies. In *IEEE VTC Fall*. IEEE.

F.H. Mirani, X. Zhang, N. Boukhatem, and T.M.T. Nguyen. 2011. Cross-layer FPS: A SCTP-based cross-layer data scheduling approach. In *IEEE CCNC*. IEEE.

E. Miyayaki and M. Oguchi. 2012. Evaluation of Middleware for Bandwidth Aggregation using Multiple Interface in Wireless Communication. *International Journal On Advances in Networks and Services* (2012).

N. Mohamed, J. Al-Jarooodi, H. Jiang, and D. Swanson. 2002. A user-level socket layer over multiple physical network interfaces. In *14th International Conference on Parallel and Distributed Computing and Systems, Cambridge, USA*.

D.D. Nguyen, Y. Xia, C. K. Yeo, and B. S. Lee. 2013. TCP Multi-homing Support in Heterogeneous Networks. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*.

A.J. Nicholson, Y. Chawathe, M.Y. Chen, B.D. Noble, and D. Wetherall. 2006. Improved access point selection. In *ACM MobiSys*. ACM.

P. Nikander, A. Gurtov, and T. R. Henderson. 2010. Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks. *Communications Surveys & Tutorials, IEEE* (2010).

C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. 2012. Exploring Mobile/ Wifi Handover with multipath TCP. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*. ACM.

Q. Peng, A. Walid, and S. H. Low. 2013. Multipath TCP algorithms: theory and design. (2013).

C. E. Perkins. 1997. Mobile ip. *Communications Magazine, IEEE* (1997).

D.S. Phatak and T. Goff. 2002. A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments. In *IEEE INFOCOM*. IEEE.

L. Popa, C. Raiciu, I. Stoica, and D.S. Rosenblum. 2006. Reducing congestion effects in wireless networks by multipath routing. In *IEEE ICNP*. IEEE.

C. Raiciu, C. Paasch, S. Barre, A Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. 2012. How hard can it be? designing and implementing a deployable multipath TCP. In *NSDI*.

A.L. Ramaboli, O.E. Falowo, and A.H. Chan. 2012. Bandwidth aggregation in heterogeneous wireless networks: A survey of current approaches and issues. *Journal of Network and Computer Applications* (2012).

P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. 2004. MAR: A commuter router infrastructure for the mobile Internet. In *ACM MobiSys*. ACM.

A. Saeed, K. Habak, M. Fouad, and M. Yousef. 2010. DNIS: a middleware for dynamic multiple network interfaces scheduling. *ACM SIGMOBILE Mobile Computing and Communications Review* (2010).

H. Sakakibara, M. Saito, and H. Tokuda. 2006. Design and implementation of a socket-level bandwidth aggregation mechanism for wireless networks. In *Proceedings of the 2nd annual international workshop on Wireless internet*. ACM.

D. Sarkar. 2006. A concurrent multipath TCP and its markov model. In *IEEE ICC*. IEEE.
S. Shailendra, R. Bhattacharjee, and S. K Bose. 2011. MPSCTP: A Simple and Efficient Multipath Algorithm for SCTP. *Communications Letters, IEEE* (2011).

S. Shailendra, R. Bhattacharjee, and S. K Bose. 2012. MPSCTP: A multipath variant of SCTP and its performance comparison with other multipath protocols. In *IEEE ICCS*. IEEE.

Z. U. Shamszaman, S. S. Ara, and I. Chong. 2013. Feasibility considerations of multipath TCP in dealing with big data application. In *IEEE ICOIN*. IEEE.

P. Sharma, S.J. Lee, J. Brassil, and K.G. Shin. 2004. Handheld routers: Intelligent bandwidth aggregation for mobile collaborative communities. (2004).

P. Sharma, S.J. Lee, J. Brassil, and K.G. Shin. 2007. Aggregating bandwidth for multihomed mobile collaborative communities. *Mobile Computing, IEEE Transactions on* (2007).

E. Shih, P. Bahl, and M.J. Sinclair. 2002. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *ACM MobiCom*. ACM.

P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. 2002. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks* (2002).

K. Sklower, B. Lloyd, G. McGregor, and D. Carr. 1996. *T. Coradetti, “ The PPP Multilink Protocol (MP).* Technical Report. RFC 1990, August.

R. Stewart, Q. Xie, and et al. October 2000. Stream Control Transmission Protocol. RFC 2960 (October 2000).

A. Tachibana and T. Hasegawa. 2012. A deployable scheme of CMT-SCTP with off-the-shelf android smartphones. In *IEEE WiMob*. IEEE.

K. Tan, J. Song, Q. Zhang, and M. Sridharan. 2006. A compound TCP approach for high-speed and long distance networks. (2006).

M. Tarique, K.E. Tepe, S. Adibi, and S. Erfani. 2009. Review: Survey of multipath routing protocols for mobile ad hoc networks. *Journal of Network and Computer Applications* 32 (2009).

C.B.S. Traw and J.M. Smith. 1995. Stripping within the network subsystem. *Network, IEEE* (1995).

L. Wang, S. Jang, and T. Lee. 2005. Redundant source routing for real-time services in ad-hoc networks. In *IEEE MASS*. IEEE.

X. Wang, Z. Feng, D. Fan, Y. Xue, and V. Le. 2009. A Segment-Based Adaptive Joint Session Scheduling Mechanism in Heterogeneous Wireless Networks. In *IEEE VTC Full*. IEEE.

Y.H. Wang, C.H. Tsai, H.Z. Lin, and C.A. Wang. 2007. Interference-aware QoS multipath routing for ad hoc wireless networks. *International Journal of Computers and Applications* (2007).

D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. 2011. Design, implementation and evaluation of congestion control for multipath TCP. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. USENIX Association.

Lisong Xu, Khaled Harfoush, and Injong Rhee. 2004. Binary increase congestion control (BIC) for fast long-distance networks. In *IEEE INFOCOM*. IEEE.

M. Xue, D. Gao, W. Su, S. Zhang, and H. Zhang. 2010. E2EMPT: A transport layer architecture for end-to-end multipath transfer. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*. IEEE.

Y. Yang, C. Zhong, Y. Sun, and J. Yang. 2010. Network coding based reliable disjoint and braided multipath routing for sensor networks. *Journal of Network and Computer Applications* (2010).

L. Yao, H. Wang, X. Sun, and Z. Zheng. 2011. An incentive mechanism based on the Bayes Equilibrium of game theory in peer-to-peer networks. In *IEEE ICECC*. IEEE.

G. Ye, T. N. Saadawi, and M. Lee. 2004. IPCC-SCTP: an enhancement to the standard SCTP to support multi-homing efficiently. In *Performance, Computing, and Communications, 2004* Proceedings of the *IEEE International Conference on*. IEEE.

M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. 2004. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *Proceedings of the annual conference on USENIX Annual Technical Conference*. USENIX Association.

D. Zhu, M.W. Mutka, and Z. Cen. 2004. QoS aware wireless bandwidth aggregation (QAWBA) by integrating cellular and ad-hoc networks. In *Proceedings of the First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*. Citeseer.

H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen. 2009. SMART: A Secure Multilayer Credit-Based Incentive Scheme for Delay-Tolerant Networks. *Vehicular Technology, IEEE Transactions on* (2009).