symmetry

Article

BATCP: Bandwidth-Aggregation Transmission Control Protocol

Ismael Amezcua Valdovinos [1,†], Jesus Arturo Perez Diaz [2,†], Luis Javier Garcia Villalba [3,*,†] and Tai-hoon Kim [4,†]

1 Facultad de Telemática, Universidad de Colima, Av. Universidad No 333-Colonia Las Víboras, Colima 28040, Mexico; ismaelamezcua@ucol.mx
2 Computer Science Department, Faculty of Engineering, Tecnológico de Monterrey, Autopista del Sol km 104 + 060, Xochitepec, Morelos 62790, Mexico; jesus.arturo.perez@itesm.mx
3 Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Computer Science and Engineering, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, Spain
4 Department of Convergence Security, Sungshin Womens University, 249-1 Dongseon-dong 3-ga, Seoul 136-742, Korea; taihoonn@daum.net
* Correspondence: javiergv@fdi.ucm.es; Tel.: +34-91-394-7638
† These authors contributed equally to this work.

Received: 21 July 2017; Accepted: 17 August 2017; Published: 21 August 2017

Abstract: The Transmission Control Protocol (TCP) is the most used transport protocol to exchange reliable data between network devices. A considerable number of extensions have been implemented into TCP to achieve better performance. In this paper, we will present, describe, implement, and analyze a new protocol extension called Bandwidth-Aggregation TCP (BATCP), which enables the concurrent use of network interfaces, to improve network performance on multi-homed nodes. BATCP allows the use of multiple TCP connections to accept multiple IP addresses from a multi-homed node, scheduling segments among them based on a scheduling algorithm. Our results show that BATCP achieves full exploitation of each network interface, achieving up to 100% network utilization using two ADSL connections in real-world scenarios. MultiPath TCP (MPTCP) is currently being standardized, and achieves up to 96% of network utilization when in ideal conditions. BATCP and MPTCP are the only protocols tested on real-world scenarios. Related work such as the Proxy Inverse Multiplexer, called PRISM, and bandwidth aggregation with Stream Control Transmission Protocol (SCTP) achieve 80% utilization or less with network simulators.

Keywords: protocol design; bandwidth aggregation; performance improvement; TCP; scheduling data; heterogeneous networks

1. Introduction

One of the key fundamental Transmission Control Protocol (TCP)/Internet Protocol (IP) architectural design principles is to build a highly flexible and reliable network capable of supporting multiple services while using a variety of link and physical layers. This is a strong reason for the adoption of the TCP/IP protocol suite as a standard for the Internet. The Transmission Control Protocol (TCP) is a connection-oriented, end-to-end reliable transport protocol designed to fit into a layered hierarchy of protocols which support multi-network applications [1].

Even though TCP was originally designed to be used over wired networks, it is now widely used in wireless environments as well. However, its sliding window and congestion avoidance mechanisms were designed to avoid router congestion instead of packet loss caused by the network [2]. Network congestion is an assumption for packet loss in many networks.

In order for TCP to overcome the problems in wireless networks, a set of extensions are used to improve its performance [3]. An extension is the implementation of a mechanism using the TCP option field that forces TCP to act differently under certain circumstances. In wireless networks, the most used extensions are maximum segment size option and timestamps, among others.

On the other hand, the rapid development and deployment of heterogeneous access networks (network technologies used to have access to the Internet) provide the opportunity for users with multi-homed devices to choose from any available network to connect to the Internet. A multi-homed device is usually a mobile device equipped with multiple heterogeneous network interfaces (different network technologies).

However, multi-homed devices do not provide mechanisms for use on multiple network interfaces in a simultaneous manner in order to improve network performance, thus restricting the full exploitation of the network resources of these devices.

In this paper, we present, describe, implement, and analyze a new TCP extension called Bandwidth-Aggregation TCP that allows the use of these multiple network interfaces simultaneously while remaining transparent to users and applications (no further modifications are required). The extension relies on the use of TCP options to provide a new capability to use multiple network interfaces while being backwards compatible with legacy TCP when one of the nodes does not support the extension.

The rest of the paper is structured as follows. Section 2 describes related work on the bandwidth aggregation problem for multi-homed devices. Section 3 describes the proposed protocol to achieve bandwidth aggregation. Section 4 shows the topology used for experimentation and compares the performance obtained by using the proposed protocol with the performance obtained by several related works. Section 5 gives conclusions and future work.

2. Related Work

Related work on bandwidth aggregation can be classified in layers as the Open Systems Interconnection (OSI) reference model is structured.

2.1. Link Layer Solutions

Link layer solutions are commonly used in the context of Web servers, mail servers, among others. They require equipment supporting the IEEE 802.3ad standard (link aggregation) [4]. Commercial and open-source implementations of link layer solutions are Cisco's EtherChannel [5], Nortel's Split Multi-Link Trunking (SMLT) [6], and GNU/Linux Bonding [7].

Link layer solutions based on the IEEE 802.3ad standard have three basic requirements: (1) the network interface cards must be compatible with the bandwidth aggregation protocol; (2) nodes with two or more network interfaces must belong to the same administrative domain; and (3) links must be homogeneous.

Although the IEEE 802.3ad standard has interesting advantages (i.e., keeps packet format intact, additional buffers are not required, and packets are sent in order), link layer solutions are not feasible in a heterogeneous network environment where different network technologies are used. Additionally, these kinds of solutions are difficult to implement and develop because they require direct access to the hardware in order to update the firmware and achieve the desired configuration.

2.2. Network Layer Solutions

Network layer solutions are based on the use of proxy servers to aggregate bandwidth. Usually, the proxy implements Mobile IP which is used for packet interception and encapsulation. In this architecture, a multi-homed device has multiple *care-of* addresses, which is a temporary IP address used for packet forwarding while the device is moving from one network to another. The server intercepts packets coming from these addresses and encapsulates them with its own IP address to forward them. The remove devices sends responses to the server, encapsulating them again with the care-of addresses of the multi-homed device. A scheduling algorithm on both the multi-homed device and the server is required in order to determine how many packets are going to be sent on each network interface.

Chebrolu et al. propose in [8–11] a network layer solution based on Mobile IP. The remote host has multiple care-of addresses, and the network proxy uses them to redirect incoming traffic from these addresses to the multi-homed node's local address. The network proxy has a scheduling algorithm that distributes packets to the remote address using an *Earliest Delivery Path First* scheme.

Network layer solutions found in [8–21] have to deal with a common constraint: the requirement of a modified proxy server based on Mobile IP. Experimental results show that Mobile IP solely introduces a significant amount of delay in the communication, and thus minimizes performance [22–24].

2.3. Transport Layer Solutions

Transport layer solutions usually involve the design and implementation of new transport protocols or the adaptation of existing protocols on each participating device. Most solutions are based on the modification of reliable transport protocols such as TCP or the Stream Control Transmission Protocol (SCTP), while other solutions propose the implementation of new transport protocols.

One particular transport layer solution is called MultiPath TCP (MPTCP) [25], and is currently an experimental standard by the Internet Engineering Task Force (IETF). It defines a set of extensions for regular TCP to allow one TCP connection to be spread across multiple paths. Multipath-aware applications are able to use an extended socket API to have further influence on the behavior of the protocol [26].

The basic operation of MPTCP is as follows:

- To a non-MPTCP-aware application, MPTCP is indistinguishable from normal TCP. All MPTCP operation is carried by the MPTCP implementation, although extended APIs could provide additional control.
- An MPTCP connection begins as a single TCP session.
- If extra paths are available, additional TCP sessions are created on these paths, and are combined with the existing session, which continues to appear as a single connection to the applications at both ends.
- MPTCP identifies multiple paths by the presence of multiple addresses at endpoints. Combinations of these multiple addresses equate to the additional paths.
- The discovery and setup of additional TCP sessions (sub-flows) will be achieved through a protocol primitive called ADD_ADDR which indicates the IP addresses of the available network interfaces.
- The exact properties of these TCP sessions that are logically bonded are dependent upon the congestion and flow control characteristics of the endpoints' MPTCP implementation.
- MPTCP adds connection-level sequence numbers in order to reassemble the data stream in-order from multiple sub-flows. Connections are terminated by connection-level *FIN* packets as well as those relating to the individual sub-flows.

In [27], the authors measure performance of MPTCP on a Software Defined Network (SDN) with two different network topologies: fat tree and jellyfish. The fat tree topology allows the use of different layers at the network level in order to aggregate bandwidth in a tree-based topology. The jellyfish topology on the other hand is constructed by taking a number of switches and randomly joining them. Hosts are then uniformly distributed among the switches. Authors used two types of traffic: discontinuous traffic (UT) and permutation traffic (PT). Results show that MPTCP with fat tree topology and PT achieved a 90.8% of network usage, while using UT achieved 65.4% of network usage. Using the jellyfish topology, MPTCP achieved 96% using PT and 67.2% using UT.

Hsieh et al. in [28–35] propose pTCP (parallel TCP), which uses a modified version of TCP called TCP-v (TCP-virtual) to open socket connections on each interface of the multi-homed node. pTCP creates and maintains TCP-v pipes for each interface. The number of pipes can be controlled by a socket option if the user does not want to use all available interfaces. Each TCP-v pipe has the same state machine and congestion control mechanism implemented in TCP. pTCP uses a scheduling algorithm based on the window size of each TCP-v pipe to distribute data across pipes. This approach is not optimal for use on wireless networks, because TCP's sliding window and congestion avoidance mechanisms were designed to avoid router congestion instead of wireless connectivity and signal strength problems [2].

Another set of transport solutions [36–43] require the modification of existing protocols such as SCTP and TCP. Both sets of solutions require the use of a scheduling algorithm to determine the amount of traffic to be sent on each network interface. Most of these algorithms base their calculations on the window size parameter. However, the sliding window and congestion avoidance mechanisms are not suitable for wireless networks because they were designed to avoid router congestion instead of packet loss. This makes the implementation of these solutions difficult to optimize on multi-homed devices such as mobile cellphones.

2.4. Cross-Layer Solutions

Cross-layer solutions involve one or more layers in their design. Solutions found in [26,44,45] require neither the modification of existing protocols nor the use of extra hardware. However, existing applications must be modified in order to open sockets from specific implementations. These solutions are efficient, but not transparent to applications.

SBAM (socket-level bandwidth aggregation mechanism) [26] is implemented at the socket layer in the operating system. A single socket is used for each connection, handling multiple network interfaces at the same time. The components of SBAM are as follows:

- The *Policy Notification Function* notifies the user policy from user to kernel space, which is a user request to use multiple network interfaces.
- The *Network Monitoring Function* collects the delay and available bandwidth of each link. Delay is monitored by sending Internet Control Message Protocol (ICMP) packets periodically to the remote host.
- The *Network Interfaces Status Notification Function* notifies the available number of network interfaces to the remote host.
- The *Send Data Schedule Function* determines the amount of data to fill the bandwidth-delay product of each link based on the delay and available bandwidth information of each connection.
- The *Send Data Division Function* divides data to send it according to the Send Data Schedule Function.
- The *Receive Data Aggregation Function* aggregates and reorders packets at the receiver.

3. Bandwidth-Aggregation TCP

The simultaneous use of multiple network interfaces is a challenging problem. Some constraints involve the partition and distribution of data across multiple interfaces while maximizing their usage, the correct reassembly of the data, the adaptation to disconnections, among others. In this paper, we propose a transport layer solution implemented as a TCP extension to the bandwidth aggregation problem.

Figure 1 depicts the modules used by BATCP to achieve bandwidth aggregation. The Connection Manager module is used to establish, maintain, open, close, and abort connections between the multi-homed node and the remote node. When a connection is established, the Data Processing module is responsible for tagging the data with a segment number used for reassembly. The Packet Distribution module determines the amount of bytes that are going to be sent on each network interface based on the user's preference and to the current performance of each interface. N_1, N_2, N_3 are the numbered network interfaces which the multi-homed node has. The protocol and scheduling algorithms were designed to use two or more network interfaces. Each network interface is connected to its respective internet service provider ISP_1, ISP_2, ISP_3. The Packet Reception module is used to receive packets from different network interfaces. This last module passes the received packets to the Packet Reassembly module where the packets are reassembled according to the segment number tagged in the Data Processing module, and the reassembled data is passed to the application layer for later processing.

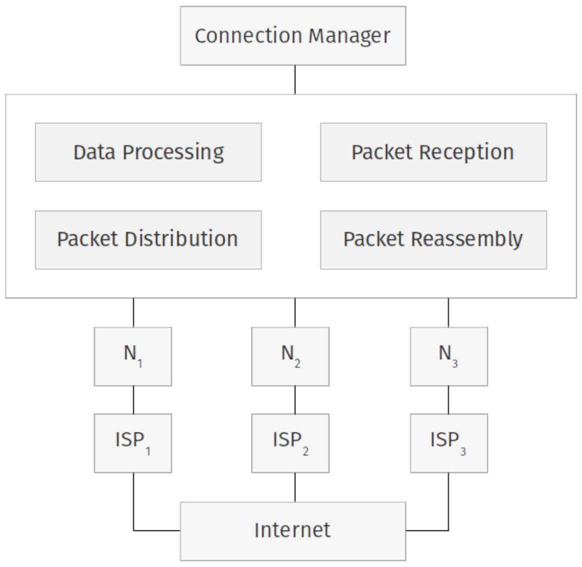

Figure 1. Bandwidth-Aggregation Transmission Control Protocol (BATCP) system diagram. ISP: internet service provider; N: network interface.

As mentioned before, a TCP extension is implemented using TCP's option field, which is structured as follows: eight octets belong to the option kind (a number that identifies the option to be used), eight octets for the option length (length in bytes of the entire option structure), and a variable length of octets for the option data.

A regular TCP connection is composed of two IP addresses and two port numbers. To create a TCP connection that uses multiple network interfaces, the connection function of the socket must be adapted in order to accept multiple IP addresses and port numbers as parameters, aggregating them and using them as a single connection. We implemented a new TCP extension that performs this task. The extension uses 254 as the value for the option kind, and the option data is composed of a *connection ID* and a *network interface number* which is specified in a six digit number composed as follows: the first two digits describe the number of the interface the multi-homed device has, and the last four digits describe the connection ID which is defined as a random number that identifies the connection for multiplexing purposes.

3.1. Opening a BATCP Connection

When the two communicating devices support the BATCP extension, the legacy three-way handshake algorithm used in legacy TCP to establish a connection is modified. The new handshake algorithm follows the exchange:

- The multi-homed device sends a synchronization segment (SYN) segment to the remote device with the BATCP extension enabled using one of its network interfaces (wired networks have priority).
- The remote device receives the SYN segment and sends back an acknowledgement (ACK) segment responding to the SYN segment sent by the multi-homed device with the BATCP extension enabled and waits for the other SYN segments to arrive from the rest of the network interfaces.

- The multi-homed device receives the ACK segment and verifies whether the remote device supports the BATCP extension. If it does, the multi-homed device sends SYN segments with the extension enabled from the rest of its network interfaces to the remote device.
- The remote device receives the SYN segments and sends back ACK segments to each of the incoming addresses.
- The multi-homed device receives the ACK segments and proceeds to create a new TCP connection using the IP address and port of the remote device and its own IP addresses and ports.

Each established connection in BATCP has unique sequence numbers from the multi-homed and remote devices. When the BATCP handshake mechanism ends, each device has $n \times 2$ sequence numbers, where n is the number of network interfaces on the multi-homed device.

When the remote device is the initiator, the multi-homed device sends an ACK segment to the remote device with the extension enabled. When the remote device receives this segment, it waits for the multi-homed device to send SYN segments from the remaining network interfaces and the algorithm proceeds as described earlier.

3.2. Closing a BATCP Connection

To close a BATCP connection, after sending all data to the remote device, the multi-homed device sends a FIN segment on each network interface when the data buffer is empty. The remote device should wait for all FIN segments coming from the multi-homed device in order to reassemble data.

3.3. Resetting a BATCP Connection

To reset a BATCP connection, any network interface can be used to send the RST segment. However, it is recommended to use a wired interface (due to reliability) to send these kinds of segments.

3.4. Retransmissions in BATCP

Whenever packet loss occurs, legacy TCP uses a selective acknowledgment (SACK) mechanism to retransmit packets. SACK allows the retransmission of only missing data instead of the transmission of series of lost segments. BATCP uses the same SACK mechanism whenever packet loss occurs. This is possible because each BATCP connection is independent, and therefore each connection has its own segment numbers to trace missing packets.

3.5. Scheduling Data across Network Interfaces

In order to send data across multiple network interfaces, a scheduling algorithm is required to determine how much data should be sent on each of the interfaces. The algorithm used in our protocol requires an estimation of the throughput of each network interface to determine the number of segments that will be sent on each interface. To perform this estimation, a Kalman filtering technique is used, which is an optimal recursive data processing algorithm that incorporates all available information to provide normalized throughput measurements.

The measurement of return trip times of messages introduces noise in the form of time spikes that increase and decrease in a rapid manner. The Kalman filter allows these noisy measurements to be softened to provide a more accurate estimation to the scheduling algorithm.

The scheduling algorithm is composed of a set of variables, which must be weighted according to predefined parameters defined by the user in order to restrict the network usage of each interface. Each variable has a percentage value that represents the importance of that variable and must be chosen in such way that the sum of all weights is one. For example, if the algorithm has two variables (namely, throughput and monetary cost), and the multi-homed device has a 4G interface and a IEEE 802.11 g interface, the user may want to use the IEEE 802.11 g interface more, as it does not cost as much as the 4G interface. Then, the weights can be distributed as follows: for the 3G interface, $weight\_throughput = 0.1$ and $weight\_cost = 0.9$; for the IEEE 802.11 g interface, $weight\_throughput = 0.9$ and $weight\_cost = 0.1$.

This ensures that the scheduling algorithm will distribute more segments to the IEEE 802.11 g interface than to the 4G interface.

The proposed scheduling algorithm has two main advantages. First, it allows the user to set their preferences in order to maximize the use of one or another network interface, as was mentioned above. Second, the amount of traffic to be sent on each network interface is determined by the end-to-end performance of each link, instead of congestion control mechanisms that were not designed for different kinds of networks (e.g., high-delay or wireless networks).

When these variables are set, a so-called *relation* function is computed, which depicts the characteristics and the usability range of each network interface. Equation (1) shows how this relation value is determined.

$$\sum_{i=1}^{n} relation_i = weight\_throughput_i \times \ln\left(\frac{1}{throughput_i}\right) + weight\_cost_i \times \ln\left(cost_i\right) \tag{1}$$

where n is the number of network interfaces, $weight\_throughput_i$ is the weight chosen for the throughput variable of the interface i, and $\ln(1/throughput_i)$ is the natural logarithm of the reciprocal of the throughput value of interface i, which is computed to normalize all parameters. $weight\_cost_i$ is the weight chosen for the monetary cost of using interface i, and $\ln(cost_i)$ is the natural logarithm of the monetary cost of using interface i. From Equation (1) and the following equations, the $\sum_{i=1}^{n}$ symbol refers to the iteration of each network interface i to compute different values instead of the sum of the computed values.

Depending on the weights and values, the relation function (Equation (1) may compute negative values. If any of the values of the relation function are negative, the equation must be recomputed as Equation (2) describes:

$$\sum_{i=1}^{n} relation_i = relation_i + |min(relation)| \times 2 \tag{2}$$

Equation (2) allows the conversion to the positive side for all values in the relation function. Once the relation function is computed, a *load index* for each interface is calculated with Equation (3):

$$\sum_{i}^{n} index_i = \frac{relation_i}{max(relation)} \tag{3}$$

The load index indicates which network interface is currently performing better. As this is a minimization problem, the network interface with the lower index value is the one offering better performance at the moment. Once the load index is obtained, the actual load for each interface is now computed as in Equation (4):

$$\sum_{i=1}^{n} load_i = \left\lceil \frac{index_i}{min(index)} \right\rceil \tag{4}$$

The load value of each interface i shows how many segments are going to be sent on each interface using a weighted round-robin scheduling scheme.

3.6. Segment Reassembly

Segments on the remote device need to be reassembled before they are presented to upper layers. In our protocol, segments are marked with a segment number, which is different from the sequence number used by TCP. It represents not the number of bytes sent, but the number of segments sent. It is an atomic 10-digit value that increases by one each time a segment is sent, which is later used for segment reassembly.

3.7. Comparison between BATCP and MPTCP

MPTCP is IETF's attempt to standardize a transport protocol capable of using multiple links simultaneously. To the best of our knowledge, the mechanisms used to achieve MPTCP's goal is comparable with BATCP's, as they both use TCP options to provide such behavior. However, there are key differences between the two protocols:

- MPTCP uses a 32-bit token for the identification of each connection. The current draft does not specify the algorithm used to generate this token. If the number of MPTCP connections are in the range of thousands, the generated token may already be in use by another connection, causing major problems in the communication process. On the other hand, BATCP identifies each connection with a structure formed with all IP addresses and ports used for that connection. For example, the multi-homed device has the following addresses: 10.1.1.1:1234 and 10.2.2.2:4321, and the remote device has the following address: 192.168.1.1:80; the resulting BATCP connection ID will be 10.1.1.1:1234,10.2.2.2:4321-192.168.1.1:80. Because BATCP uses the address and port from the remote device, is it impossible to have repeated IDs.
- MPTCP uses an MP_JOIN option to aggregate new links to the MPTCP connection. BATCP passively waits for incoming connections because the remote host knows the number of network interfaces that the multi-homed device is willing to use. This number is shared between the multi-homed device and the remote device in the first SYN segment sent to initialize the connection.
- In order to reassemble segments in the remote device, MPTCP uses another TCP option called data sequence signal (DSS), which indicates the range of bytes each subflow is willing to transfer (must indicate the beginning and the end of the bytes as sequence numbers). BATCP determines the amount of segments to be sent on each interface by using the scheduling algorithm. The size of each segment may change if the network conditions of the network interface changes using legacy TCP congestion control techniques. The protocol assigns a BATCP sequence number for later reassembly. We believe that BATCP allows more flexibility, since it adjusts the number of segments to be sent on each network interface according to the current network conditions. On the other hand, MPTCP assigns a fixed size of bytes to send on each interface a priori.

4. Experimental Tests and Results Comparison

Performance of the BATCP extension is evaluated through an implementation using GNU/Linux Ubuntu 13.10. In order to implement the protocol, the TCP/IP protocol suite for the Linux kernel was modified.

There are two experimental setups. In the first scenario (Figure 2), a multi-homed device is connected to a remote server with two ADSL connections (through an IEEE 802.11 g interface and a FastEthernet interface). In the second scenario (Figure 3, a multi-homed device is connected to a remote server with one HSPA+ network interface and one ADSL through FastEthernet interface.

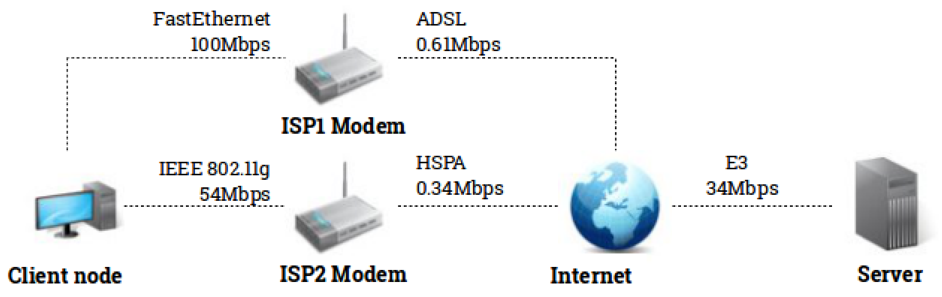

Figure 2. Experimental testbed with two ADSL connections.

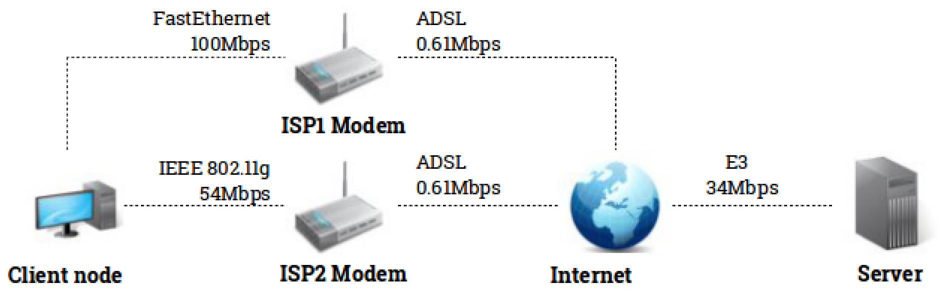

Figure 3. Experimental testbed with one ADSL connection and one HSPA+ connection.

To retrieve information about the network characteristics, we gathered return trip time measurements from the socket to feed the Kalman filter.

Implementation and performance evaluation of BATCP were conducted first by getting the approximate bandwidth of each network interface and then by a series of experiments involving the client and server devices. The experiments consisted of repeatedly sending large files (5 MB and 10 MB) through different network interfaces, evaluating the performance on each interface. Through experimentation, we found that in order to obtain accurate measurements of network performance it is required to send each file at least 25 times. The variations in throughput are not significant between each test when the number of experiments is met.

It is worth mentioning that the performance of the scheduling algorithm is directly related to the performance of the overall protocol, since the algorithm is dictating the number of segments to be sent on each network interface according to the current transmission metrics of each link. If any link is performing badly, the algorithm will schedule fewer segments on that interface.

Packet loss and error rate were not computed, since BATCP uses legacy TCP to transport data. Geoff Huston [46] investigated the impact of packet loss and error rate on TCP performance, and explains how the Slow Start and Congestion Avoidance algorithms are used in TCP to overcome these issues.

The characteristics of the client node are a Genuine Intel CPU with two 1.73 GHz processors, 1 GB RAM, 100 GB HDD, an Intel Corporation PRO/Wireless 3945abg wireless interface and an Intel Corporation PRO/100 VE wired interface. It runs Ubuntu 12.04.1 LTS GNU/Linux distribution with the 3.2.0-33 Linux kernel as operating system. The network characteristics of the client node are: a FastEthernet network card connected to an ADSL modem (ADSL1 from now on) with an uploading speed of approximately 633 Kbps, and an HSPA network card connected via IEEE 802.11 g to a modem with an uploading speed of approximately 355 kbps. As mentioned earlier, these values were calculated by sending a 5 MB and a 10 MB file 25 times on each network interface.

The network technologies used in this paper were selected with the following criterion: we wanted to perform uninterrupted tests on each technology, which was only possible at a home network at the time. We strongly believe that the algorithm and overall protocol performs accordingly when using fast or slow network interfaces, since it determines the amount of traffic for each network interface based on the current performance of said interface instead of fixed parameters. Section 5 describes future plans for enhancing the scheduling algorithm and the communication technologies to provide results using newer network technologies.

4.1. Single Interface Performance

In order to provide a reference, several tests were conducted using a single network interface at a time by sending different file sizes for throughput evaluation of each interface.

The results shown in Figure 4 indicate that the ADSL1 and ADSL2 links are stable connections, showing minimal differences among them in maximum and minimum throughput values, obtaining throughputs between 77.6726 kB/s and 79.1615 kB/s. On the other hand, the HSPA modem connection depicts an unstable network with high burst packet transferring and higher RTT and jitter measurements, obtaining values between 39.0112 kB/s and 44.3763 kB/s.

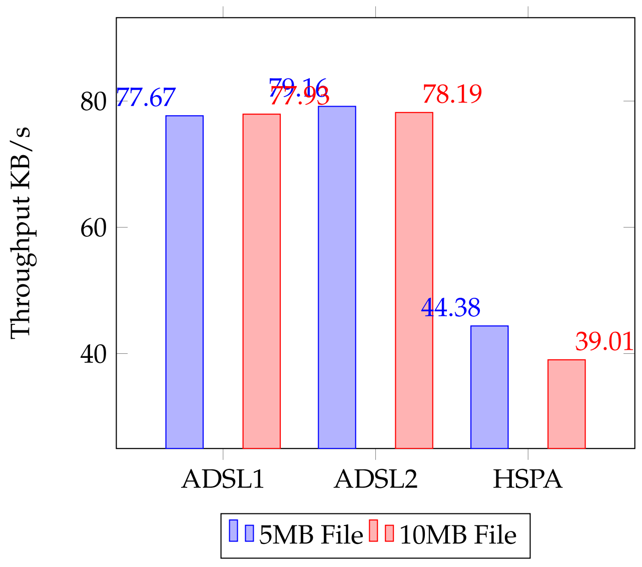

Figure 4. Single interface performance.

4.2. Multiple Interface Performance

The main purpose of BATCP is the simultaneous use of multiple interfaces. Therefore, another set of tests were conducted using two network interfaces simultaneously by sending two different file sizes to the service device.

As can be seen in the results in Figure 5, when the ADSL1 and ADSL2 interfaces were used simultaneously, the achieved throughput was approximately the sum of the performance of each individual interface (100% and 98.1% utilization, respectively). However, when the ADSL1 and the HSPA connections were used, results show poor performance. This may be caused by the use of TCP in 3G+ networks where the sliding window mechanisms freeze until acknowledgement segments are received. Additionally, timeouts and retransmissions occur because TCP thinks this is due to congestion.

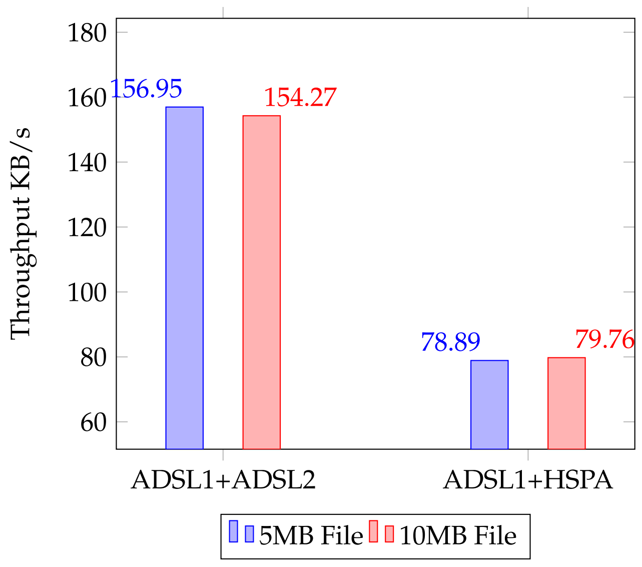

Figure 5. Multiple interface performance.

4.3. Performance Evaluation

In our related work section, every solution presented uses different connection characteristics, resulting in different final throughputs. To test the performance of BATCP, we used a utilization percentage based on the throughput obtained by each solution and the expected throughput, which is the sum of the maximum performance of each network interface. Figure 6 shows the results obtained by related solutions using the metric explained earlier. Zannettou et al. [27] evaluated the performance of MPTCP in an SDN-based data-center. However, the document does not describe network characteristics, showing only performance of the protocol based on percentage of aggregated link utilization. Therefore, MPTCP results are not shown in Figure 6.

Figure 7 shows the throughput obtained by combining different network interfaces and file sizes. BATCP1 refers to the combination of the ADSL1 and ADSL2 interfaces with a 5 MB file size; BATCP2 refers to the combination of the ADSL1 and ADSL2 interfaces with a 10 MB file size; BATCP3 refers to the combination of the ADSL1 and HSPA interfaces with a 5 MB file size; and BATCP4 refers to the combination of the ADSL1 and HSPA interfaces with a 10 MB file size.

From Figure 8, we observe that BATCP performs optimally when two stable network interfaces are used simultaneously, reaching an exploitation rate ranging from 98% to 100%. By stable connection, we are referring to networks with low jitter and delay. However, performance decreases rapidly if we use one stable and one unstable connection. A possible reason for this is the sliding window mechanisms used in TCP, which was not designed with the characteristics of wireless networks. As mentioned earlier, when TCP loses segments in a wireless environment, the protocol assumes this was due to network congestion, reducing the sliding window.

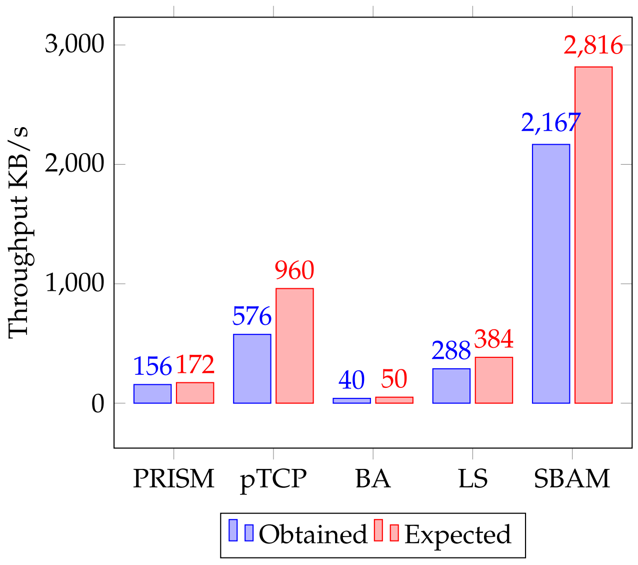

Figure 6. Obtained vs Expected results in related solutions. PRISM: Proxy Inverse Multiplexing [15,16]; pTCP: parallel TCP [28–30]; BA: Bandwidth aggregation with SCTP [36]; LS: Load-Sharing SCTP [38,39]; SBAM: socket-level bandwidth aggregation mechanism [44].

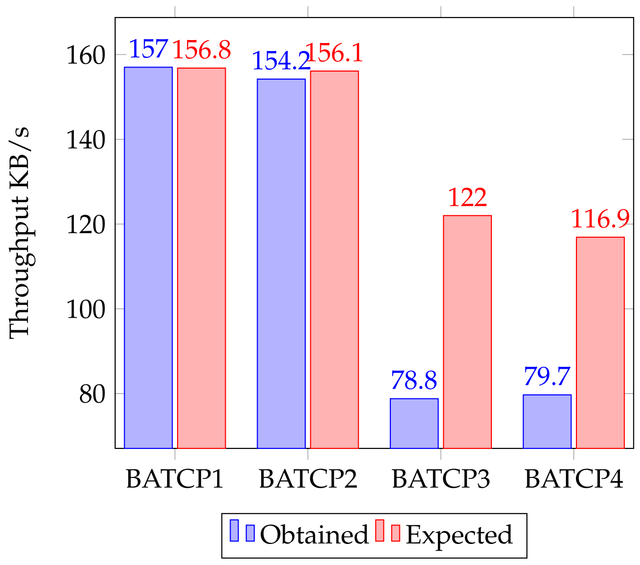

Figure 7. BATCP throughput comparison.

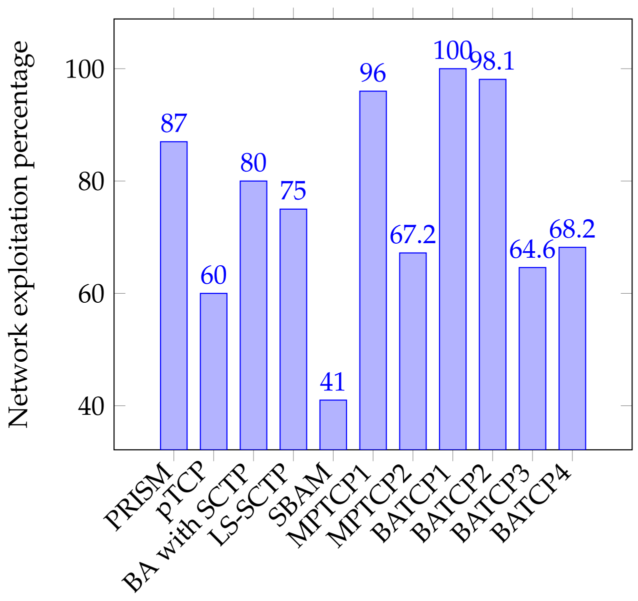

Figure 8. Performance evaluation. MPTCP: MultiPath TCP; SCTP: Stream Control Transmission Protocol.

Comparing our results to those achieved by PRISM [15,16] (high tier), BATCP has a 13% higher exploitation rate. This means that the bandwidth aggregated by BATCP has higher throughput. Another solution that shows acceptable performance is called BA with SCTP [36], with an 80% exploitation rate. However, this approach limits users to the use of the Stream Control Transmission Protocol (SCTP), which is not used by most of today's applications. The protocol called Load-Sharing SCTP (LS-SCTP) [38,39] has lower exploitation rate than Bandwidth-Aggregation with SCTP, and also restricts users to the use of SCTP.

In Figure 8, MPTCP1 and MPTCP2 are the results obtained in [27] using a jellyfish topology and permutation traffic (MPTCP1) and discontinuous traffic (MPTCP2). Results show that MPTCP performs optimally when using a constant traffic pattern, achieving 96% of network usage aggregating six MPTCP flows. On the other hand, MPTCP2 achieves 67.2% of network usage when discontinuous traffic was transferred. This result is comparable to BATCP3 and BATCP4, where the traffic using the HSPA network interface also behaves in burst mode, achieving 78.8% and 79.7% network utilization, respectively.

The low tier of proposed solutions comprises pTCP [28–30] and SBAM [44] solutions, with an exploitation rate of 60% and 41%, respectively. In BATCP, when one stable network is used in combination with a network with high jitter and delay, it achieves an exploitation rate ranging from 64% to 68%, outperforming prior solutions even when is performing poorly.

5. Conclusions

In this paper, we designed and implemented a functional protocol that enables the use of multiple network interfaces simultaneously to exchange data concurrently.

According to our experimental tests, BATCP outperforms other similar solutions to the bandwidth aggregation problem at all layers. It is worth noting that the results were obtained by an implementation of the protocol using real-world infrastructure, which introduces performance factors that simulators fail to consider. The performance of the protocol is shown in terms of a percentage usage to normalize results obtained by related solutions. BATCP achieved an exploitation rate of 100% when two network interfaces with stable performance were used. However, when we used a stable interface and a non-periodic unstable interface (e.g., the ADSL1 + HSPA+ interfaces), results were acceptable, reaching an exploitation rate of 68.2%. This may be caused by the fact that HSPA+ networks are burst-oriented technologies, causing TCP's sliding window mechanism to freeze at certain periods of time. Additionally, the quality of service (QoS) class assigned to data traffic in HSPA has the lowest priority, thus having an important impact on BATCP's performance.

Similar solutions to the bandwidth aggregation problem achieved an exploitation rate ranging from 60% to 96%. MPTCP is currently being standardized by the IETF, and is the solution that achieves optimal performance, reaching a 96% network utilization when optimal conditions are met (six sub-flows and stable connections). Additionally, MPTCP and BATCP are the only two protocols that are actually implemented in GNU/Linux, allowing tests to be performed in real-world scenarios instead of using network simulators. The solution proposed in this paper outperformed all other solutions, achieving an exploitation rate of 100% network utilization.

In future work, we will design and implement an improvement to the scheduling algorithm to overcome the problems encountered in HSPA+ networks. We will use 4G networks and other types of network traffic to obtain a better QoS class and improve delay issues arising through the experimental tests of the protocol. The use of more than two network interfaces will be tested. Additionally, a more optimal buffer size to obtain better performance will be determined. Full mobility support is also expected in future implementations of BATCP.

Author Contributions: All authors in this paper contributed equally.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Postel, J. *Internet Protocol—DARPA Internet Program Protocol Specification*; *Request For Comments 791*; Internet Engineering Task Force: Fremont, CA, USA, 1981.
2. Perez, J.A.; Donnet, B.; Bonaventure, O. Preliminary analysis of the TCP behaviour in 802.16 networks. In Proceedings of the 1st WEIRD Workshop on WiMAX, Wireless and Mobility, Coimbra, Portugal, 23–25 May 2007.
3. Jacobson, V.; Braden, R.; Borman, D. *TCP Extensions For High Performance*; *Request For Comments 1323*; Internet Engineering Task Force: Fremont, CA, USA, 1992.
4. IEEE. *Amendment to Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications—Aggregation of Multiple Link Segments*; IEEE Std. 802.3ad-2000; Lan Man Standards Committee: New York, NY, USA, 2000.
5. Cisco. Understanding EtherChannel Load Balancing and Redundancy on Catalyst Switches. Available online: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3750x_3560x/software/release/12-2_55_se/configuration/guide/3750xscg/swethchl.html#18140 (accessed on 21 May 2017).

6. Nortel. Split Multi-Link Trunking Ethernet Routing Switch 8600. Available online: http://downloads.avaya.com/css/P8/documents/100123976 (accessed on 21 May 2017).
7. Davis, T. Linux Bonding. Available online: http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding (accessed on 21 May 2017).
8. Chebrolu, K.; Rao, R. Bandwidth aggregation for real-time applications in heterogeneous wireless networks. *IEEE Trans. Mob. Comput.* **2006**, *5*, 388–403.
9. Chebrolu, K.; Rao, R. Selective frame discard for interactive video. *IEEE Int. Conf. Commun.* **2004**, *7*, 4097–4102.
10. Chebrolu, K.; Rao, R. Communication using multiple wireless interfaces. *Wirel. Commun. Netw. Conf.* **2002**, *1*, 327–331.
11. Chebrolu, K.; Raman, B.; Rao, R. A network layer approach to enable TCP over multiple interfaces. *Wirel. Netw.* **2005**, *11*, 637–650.
12. Liebsch, M.; Le, L. Simultaneous binding extension to proxy mobile IPv6 as service enabler for multi-mode mobile devices. In Proceedings of the IEEE GLOBECOM Global Telecommunications Conference, New Orleans, LO, USA, 30 November–4 December 2008.
13. Taniguchi, N.; Aust, S.; Takizawa, Y.; Yamaguchi, A.; Obana, S. Packet allocation for efficient use of multiple wireless links in cognitive radio networks. In Proceedings of the First International Global Information Infrastructure Symposium, Marrakech, Morocco, 2–6 July 2007; pp. 27–34.
14. Taleb, T.; Fernandez, J.; Hashimoto, K.; Kato, N. A bandwidth aggregation-aware QoS negotiation mechanism for next-generation wireless networks. In Proceedings of the Global Telecommunications Conference, Washington, DC, USA, 26–30 November 2007; pp. 1912–1916.
15. Kim, H.; Shin, K. Improving TCP performance over wireless networks with collaborative multi-homed mobile hosts. In Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, Seattle, Washington, DC, USA, 6–8 June 2005; pp. 107–120.
16. Kim, H.; Shin, K. Prism: Improving the performance of inverse-multiplexed TCP in wireless networks. *IEEE Trans. Mob. Comput.* **2007**, *6*, 1297–1312.
17. Huang, H.; Cai, J.; Kassler, A.; Fu, C. Load-sharing in wireless multi-homed systems. In Proceedings of the ICC 2005: 2005 IEEE International Conference on Communications, Seoul, Korea, 16–20 May 2005; Volume 5, pp. 3489–3493.
18. Wasserman, M. Current Practices for Multiple Interface Hosts. Available online: http://tools.ietf.org/html/draft-ietf-mif-current-practices-00 (accessed on 21 May 2017).
19. Sharma, P.; Lee, S.; Brassil, J.; Shin, K. Handheld routers: Intelligent bandwidth aggregation for mobile collaborative communities. In Proceedings of the 1st International Conference on Broadband Networks, San Jose, CA, USA, 25–29 October 2004; pp. 537–547.
20. Sharma, P.; Lee, S.; Brassil, J.; Shin, K. Aggregating bandwidth for multihomed mobile collaborative communities. *IEEE Trans. Mob. Comput.* **2007**, *6*, 280–296.
21. Rodriguez, P.; Chakravorty, R.; Chesterfield, J.; Pratt, I.; Banerjee, S. MAR: A commuter router infrastructure for the mobile internet. In Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services, Boston, MA, USA, 6–9 June 2004; pp. 217–230.
22. Samad, M.; Herman, S. Quality of service for mobile IP services in wireless network. In Proceedings of the Asia-Pacific Conference on Applied Electromagnetics, Johor, Malaysia, 20–21 December 2005.
23. Best, P.; Pendse, R. Quantitative analysis of enhanced mobile IP. *IEEE Commun. Mag.* **2006**, *44*, 66–72.
24. Samad, M.; Kasim, N. Effect of transmission delay and tunneled traffic in a wireless mobile IP network. In Proceedings of the RF and Microwave Conference, Putra Jaya, Malaysia, 12–14 September 2006; pp. 398–401.
25. Ford, A.; Raiciu, C.; Handley, M.; Barre, S. TCP Extensions for Multipath Operation with Multiple Addresses. Available online: http://tools.ietf.org/html/draft-ford-mptcp-multiaddressed-01 (accessed on 21 May 2017).
26. Liu, D.; Cao, Z. Socket API Extension for Multiple Connection Support. Available online: http://tools.ietf.org/html/draft-liu-mif-api-extension-01 (accessed on 21 May 2017).
27. Zannettou, S.; Sirivianos, M.; Papadopoulos, F. Exploiting path diversity in datacenters using MPTCP-aware SND. In Proceedings of the IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 27–30 June 2016; pp. 539–546.
28. Hsieh, H.; Sivakumar, R. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. In Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, Atlanta, GA, USA, 23–28 September 2002; pp. 83–94.

29. Hsieh, H.; Sivakumar, R. pTCP: An end-to-end transport layer protocol for striped connections. In Proceedings of the 10th IEEE International Conference on Network Protocols, Paris, France, 12–15 November 2002; pp. 24–33.
30. Hsieh, H.; Kim, H.; Sivakumar, R. An end-to-end approach for transparent mobility across heterogeneous wireless networks. *Mob. Netw. Appl.* **2004**, *9*, 363–378.
31. Magalhaes, L.; Kravets, R. Transport level mechanisms for bandwidth aggregation on mobile hosts. In Proceedings of the 9th International Conference on Network Protocols, Riverside, CA, USA, 11–14 Novermber 2001; pp. 165–171.
32. Keshav, S. A control-theoretic approach to flow control. *SIGCOMM Comput. Commun. Rev.* **1995**, *25*, 188–201.
33. Park, K.; Park, D. Negotiation-based transport layer protocol for many-to-one transmission. In Proceedings of the IEEE 10th International Symposium on Consumer Electronics, St. Petesburg, Russia, 28 June–1 July 2006; pp. 1–5.
34. Kim, H.; Zhu, Y.; Sivakumar, R.; Hsieh, H. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. *Wirel. Netw.* **2005**, *11*, 363–382.
35. Hsieh, H.; Kim, K.; Zhu, Y.; Sivakumar, R. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, San Diego, CA, USA, 14–19 September 2003; pp. 1–15.
36. Argyriou, A.; Madisetti, V. Bandwidth aggregation with SCTP. In Proceedings of the Global Telecommunications Conference, San Francisco, CA, USA, 1–5 December 2003; pp. 3716–3721.
37. Huang, C.; Zeng, Z. SCTP-based bandwidth aggregation across heterogeneous networks. *Comput. Intell. Ind. Appl.* **2008**, *1*, 650–654.
38. Abd, A.; Saadawi, T.; Lee, M. Bandwidth aggregation in Stream Control Transmission Protocol. In Proceedings of the 9th International Symposium on Computers and Communications, Alexandria, Egypt, 28 June–1 July 2004; Volume 2, pp. 975–980.
39. El Al, A.; Saadawi, T.; Lee, M. A transport layer load sharing mechanism for mobile wireless hosts. In Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops, Orlando, FL, USA, 14–17 March 2004; pp. 87–91.
40. Anand, J.; Sarkar, D. Architecture, implementation, and evaluation of a concurrent multi-path real-time transport control protocol. In Proceedings of the Military Communications Conference, Orlando, FL, USA, 29–31 October 2007; pp. 1–7.
41. Sarkar, D.; Paul, S.; Narayan, H.; Sarkar, U.; Prasad, S. Effect of path parameter imbalance on the performance of concurrent multipath TCPs. In Proceedings of the Military Communications Conference, Orlando, FL, USA, 29–31 October 2007; pp. 1–6.
42. Nguyen, H.; Bonnet, C. Enhancements for simultaneous access in network-based localized mobility management. In Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, Athens, Greece, 3–7 September 2007; pp. 1–5.
43. Nguyen, H.; Bonnet, C. An intelligent tunneling framework for always best connected support in network mobility (NEMO). In Proceedings of the Wireless Communications and Networking Conference, Las Vegas, NV, USA, 31 March–3 April 2008; pp. 3021–3026.
44. Sakakibara, H.; Saito, M.; Tokuda, H. Design and implementation of a socket-level bandwidth aggregation mechanism for wireless networks. In Proceedings of the 2nd Annual International Workshop on Wireless Internet, Boston, MA, USA, 2–5 August 2006; p. 11.
45. Balakrishnan, M.; Mishra, R.; Rao, R. On the use of bandwidth aggregation over heterogeneous last miles. In Proceedings of the 2nd International Conference on Broadband Networks, Boston, MA, USA, 3–7 October 2005; Volume 2, pp. 1541–1547.
46. Huston, G. TCP Performance. Available online: http://www.cisco.com/warp/public/759/ipj_3-2_tcp.html (accessed on 21 May 2017).