Adaptive Session Duration for Efficient Pooling of Time and Space in Content Delivery

Mohammad Malli† **Hassan Sbeity†** **Ahmad Fadlallah†** **Ali Hodroj††** **Abed-Ellatif Samhat ††**

†Faculty of Computer Studies, Arab Open University, Lebanon

††Faculty of Engineering, Lebanese University, Lebanon

Summary
The availability of a wide variety of access technologies provides end- users (clients) with access to more than one communication network. In addition, service/content replication has become a trivial approach in overlay networks to provide a high availability of data and better Quality of Service (QoS). In this paper, we consider a multi-homed client seeking a replicated service/content in a Content Distributed Network (CDN), and we propose an adaptive model to improve the content distribution. This model is applied at the application level in a fully distributed way. It takes advantage of the connections "multiplicity" at both client and server sides in order to improve the content distribution. It consists of simultaneously downloading the requested content from the "best" server available through each client network interface. This is achieved in multiple sessions having adaptive durations estimated based on the passive measurement of network performance. In order to ensure the confidentiality of the content download, we enhance the model with an encryption scheme. Our measurement results show that our model is able to pool efficiently the network capacity over the space and time by improving considerably the latency.

Key words:
Multihoming, Service Replication, Passive Measurement, Content Delivery, Confidentiality

1. Introduction

Most End-user devices (PCs, Laptops, Handheld devices, etc.) are equipped nowadays with multiple network interfaces that allow users to access different network technologies whether wired (Ethernet, Fiber, etc.) or wireless (Wi-Fi, WiMAX, LTE, etc.), which is known as "multi-homing". The advantages of multi-homing, from current users' perspective, are: (1) the existence of alternative solutions (e.g. laptop users using Wi-Fi in the absence of Ethernet network) and (2) the use of each technology for specific purpose(s) (e.g. Mobile phone users using Bluetooth for file sharing and using LTE or Wi-Fi for Internet access). However, there is a growing interest in exploiting client multi-homing in another way through the simultaneous use of different network interfaces for the same service. This interest is driven by the possibility to access the Internet using different available network technologies: Wi-Fi, LTE, Ethernet, Bluetooth (in case a Bluetooth device is designated as an access point), and by the users' interest to take advantage of all the available resources and to improve their quality of experience (QoE) and quality of service (QoS). A typical example of application (from an end-user perspective) of multi-homing in such context is file download applications allowing to download the same file from the same server using different network interfaces. In a related domain, service/content replication is a scalable solution for the distribution of digital content over the Internet. It mainly aims to improve the quality of service (users can access the content from the "best"1 servers hosting the requested content) and to ensure high data availability. Service replication can be achieved through different proposed overlay networks such as Content Distribution Networks (CDN) (e.g. Akamai [1]).

In this work, we investigate the use of service replication in file downloading for a multi-homed network. A typical example is a client with multiple network interfaces downloading a content replicated in different servers of a CDN. The content requested by a client can be hosted on different servers. For a better QoS, the "best" server to request the file from, should be selected. The best server is defined as the one capable of providing the requested content to the client with the highest QoS compared to other servers (providing the requested content). The best server is defined per each network interface; each client can have a different best server for each network interface based on the conditions on the path connecting the client to the different servers through each network interface. The same goes between different clients for the same content; a client might have a best server different from the best servers of other clients.

The objective of this work is to optimize (minimize) the total download time of a file replicated in the mirror servers of one CDN network or in multiple CDN networks (i.e., content multi-homing [18] [3] [11]). In order to achieve this objective, an application-level protocol is proposed. It works as follows: a client is

1 providing the higher download bitrate for users

Manuscript received February 5, 2016
Manuscript revised February 20, 2016

downloading a replicated content from a certain set of best mirror servers simultaneously through its available network connections. First, the best server reachable through each network interface is determined based on any application utility function (e.g., [20] [21]). Then, the requested content is downloaded from these best servers simultaneously but with different calculated amounts. File chunks are of variable size in order to deal with the dynamicity of the network performance status. Content download is partitioned into successive sessions such that the network performance might considerably change after each session time. The session duration is estimated based on the passive measurement of the network performance. The security of the content download is also addressed by shuffling and encrypting the chunks.

The rest of the paper is organized as follows: Section 2 briefly presents the related works. Section 3 describes the proposed model in details. Section 4 presents the security part of this model. Section 5 presents the evaluation of the proposed solution through emulated scenarios. Finally, section 6 concludes the paper.

2. Related Work

The proposed solution falls in the following three domains: multi-homing, multi- path and service replication. Our survey of the similar works shows that - to the best of our knowledge - no solution covers all these aspects together, although some solutions include two of them. Nevertheless, the aim of the proposed solution is to optimize the latency in this context by proposing an adaptive model for download sessions.

Many solutions [24] [4] [25] [26] were proposed to enhance the content distribution; they rely on particular network infrastructure nodes (e.g. load-aware Anycast router, route controller, peer coordinator, etc.) taking into account network-specific constraints (e.g. traffic engineering constraints) perceived by the Internet Service Provider (ISP) or the overlay network operator in order to solve the problem as a global optimization problem.

While such approaches could be of great benefit for traffic engineering purposes, end-systems solutions are able to provide better enhancement of the performance perceived by the clients. In some scenarios, end-systems solutions are able to achieve better traffic engineering outcome than the ISPs themselves can do as shown in [29].

Furthermore, deployment-related issues of the existing solutions can be avoided by implementing the solution at the end-user level in a fully distributed way. This high interest in multi-path-capable end systems has been demonstrated in several research works [29] [30] [13] [31] [14] in addition to the ongoing work of standardization bodies [10]. However, there is no standardized transport protocol with such capability so far.

This work proposes to determine the best mirror server reachable through each client network interface using any application utility function (e.g., [20] [21]). Then, the client downloads the requested content from the determined best servers simultaneously through their corresponding interfaces. Each best server should deliver a specific estimated range of bytes (i.e. content chunk) to an independent TCP socket opened at the client side for being finally aggregated at the application-level. This is achieved in multiple sessions where in each session the size of each chunk is estimated based on the current network performance which is estimated passively. The proposed model has also taken advantage of this content delivery approach (multiple connections with multiple sessions) for strengthening the confidentiality of data transfer.

3. Model Description

The aim of this work is to offer a solution that optimally utilizes the multi-homing capabilities of a client wishing to download a content replicated on different CDN servers. The model is illustrated by considering a client requesting a content replicated in different servers. The client - through this model - is capable of downloading the content simultaneously and efficiently from the different servers using its available network connections. The simultaneous download requires dividing the content to be downloaded into a number of chunks; a chunk is to be downloaded from one of the mirror servers. The efficiency of content download is achieved by (1) selecting the best server for each network connection and avoiding the overlap and redundancy in downloaded chunks, and (2) considering different chunk sizes for each connection since the performance of the different available network connections are not the same; the throughput achieved through some connection(s) can be largely greater than that achieved through other connections.

The best server for each network connection is determined, by the client, before starting the download using selection techniques such as that described in [21]. The selection process can also be triggered at a certain interface when the current corresponding best server is down. It can also be triggered when this best server is underperforming, however, this introduces additional overhead and thus might be deprecated. After selecting the best servers, the content download starts; it consists of one startup session and several download sessions running simultaneously through the different network connections. The duration of each download session is calculated using a session duration estimation algorithm (presented in 3.1). The size of chunks to be downloaded in a session through the

corresponding network connection is calculated based on the throughput achieved in the previous session.
Both calculations (chunk size and session duration) are performed during a control phase that precedes each download session (as shown in Figure 1).

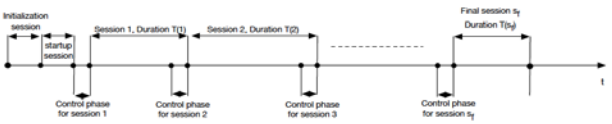


Figure 1- Content Download Sessions

The chunk size for the first session (called the startup session) is fixed (65 Kilobytes 2) for all network connections. The time taken to download this chunk is used to estimate the throughput at each interface, which will serve to calculate the chunk size for the first "regular" download session. Before proceeding with the detailed model description, Table 1 summarizes the different terms and notations used in the rest of this paper.

Table 1-Terms and Notations

C	The client requesting the content in the CDN network	T_{min}	Minimum session duration
n	The number of servers holding the content in the CDN network	T_{max}	Maximum session duration
$i=1..n$	Index of servers holding the content to be downloaded	R_{th}	Throughput variation threshold
Si	The servers holding the content in the CDN network	$Th(S_i)=(Th_1(s_k))$	Estimated throughput matrix for the client during a session s_k
l	Number of network interfaces/connections of the client C^3	$Th_j(s_k))$	Estimated throughput on the path between the client and server S_i through network interface c_j during a session s_k
$j=1..l$	Index of network interface/connections for the client C	$Th_j(s_k)$	Vector of throughputs achieved during s_k for the different network connections
c_j	Network interfaces/connections for the client C	$Th_{max}(s_k)$	$Th_{max}(s_k)= MAX_{i=1,l}$ $(Th_i(s_k))$: Maximum throughput achieved during a session s_k among all the network interfaces
$k=1..f$	Index of content download sessions	$BS(s_k)=(BS(s_k))$	Vector of best servers per network connection during a content download session s_k
s_k	Content download session	$BS_j(s_k)$	The server that satisfies $MAX_i(Th_i(s_k))$ during a session s_k for a content download session s_k
s_f	Final content download session	$CS(s_k)$	Chunk size to be downloaded by the client through its network connection c_j during a session s_k
$T(s_k)$	Duration of the session s_k		

The chunk size (CSj (sk)) at a session (sk) for a network connection (cj) is calculated as follows:

$$\boldsymbol{CS_j(s_k) = Th_j(s_{k-1}) * T(s_k)} \quad (1)$$

where (Thj (sk-1)) is the average TCP throughput at the previous session (sk-1) through the network connection (cj) and (T(sk)) is the estimated duration for session sk (calculated using the session duration estimation algorithm presented in section 3.1). The last session sf is reached when the expected number of bytes to be downloaded (sum of all chunk sizes) is greater than the number of remaining bytes R to be downloaded from the content:

$$\sum_{j=1}^{n} \boldsymbol{CS_j(s_f) > R} \quad (2)$$

2 TCP maximum buffer size

The chunk size during the last session (sf) through a network connection cj is set proportionally to the relative throughput according to Equation 3:

$$\boldsymbol{CS_j(s_f) = R * \left(\left(\frac{\frac{Th_j(s_{f-1})}{Th_{max}(s_{f-1})}}{\sum_{j=1}^{l} \left(\frac{Th_j(s_{f-1})}{Th_{max}(s_{f-1})} \right)} \right) \right)} \quad (3)$$

where, Thmax (sk) is the maximum throughput achieved in the session (sk) among all the network connections:

$$\boldsymbol{Th_{max}(s_k) = MAX_{j=1...l} Th_j(s_k)} \quad (4)$$

If the proportions calculated in Equation 3 do not sum up to the remaining bytes (R), the residual bytes are added to the range of bytes allocated to the best server (associated to the interface achieving the maximum throughput during the last session). Finally, the client application, upon reception of the different chunks, re-sequence the received chunks using a buffer of the same size of the requested content.

3.1 Session duration estimation algorithm

The main rationale for varying the session duration is to respond to the dynamicity of network performance. However, it is not feasible to find optimal values for the session duration T (sk) when running the scenarios over the internet; one cannot expect the values of time intervals during which changes of bandwidth might occur on the paths between client and servers. Basically, network performance can change (improvement/degradation) considerably as a result of traffic variability as well as routing changes. This variation has been addressed in several research works.
[17] showed that routing events usually converge within several minutes. Besides, the authors in [15] showed that bottlenecks are more likely to occur on inter-AS (Autonomous System) links which could be changed in time depending on the level of congestion between the connected ASs. According to [19], considerable changes in bandwidth are detected every 20 minutes. Moreover, the measurements presented in [12] showed that approximately two-thirds of all bandwidth detours persist for more than 90 minutes. In order to address the problem of unpredictable dynamicity of network performance, a dynamic value of T(sk) in each session sk is selected, which is estimated according to the algorithm presented below:
- Session 0 (Startup Session): The first session is denoted as a startup session s0. It consists of downloading (for each connection) 65 Kilobytes of content data. The reason is that at this stage, rapid and passive measurement of the average TCP throughput is needed in order to estimate an optimized value of the chunk size for the next sessions. It is important to mention that the choice for

passive measurement avoids the overhead of injecting active measurement packets.

- Session 1: The passive measurement of the average throughput (achieved in s0) can be used to estimate the chunk size of session s1 as per Equation 1. However, it is not sufficient to estimate the dynamicity of network performance and subsequently the optimal value of T(s1). Then, the period of this session is set to its minimum value Tmin where most likely the network performance will not considerably change before Tmin.
- Session k ($k \geq 2$): The session estimation algorithm for session k is illustrated in Figure 2. If the average throughput in the last two sessions increased/decreased by less than a preset threshold ratio Rth, then it is assumed that there is no considerable change in network performance during the current session duration. Subsequently, the session duration is doubled for the next session checking (until reaching a preset maximum duration Tmax). This will reduce the number of the control phases and thus minimize their overheads (compared to shorter values of session period). On the other hand, if the average throughput in the last two sessions increased/decreased by more than Rth, a considerable change in network performance during the current session duration is assumed. Hence, the next session duration is decreased to the half, in order to have sessions where the network status remains steady. Finally, the session duration is set to the minimum value between the session durations estimated on every network connection (i.e., joining a network interface to its associated server) to synchronize the transfer of chunks among the different network connections. A tradeoff is needed for setting Tmin and Tmax: choosing longer time interval for the download sessions (larger Tmax) may lead to less optimal estimation of the chunks sizes but it reduces the overhead of the control phases. The opposite will happen when choosing shorter time intervals (smaller Tmin) where the overhead of the control phase increases for the benefit of more optimized session duration.

$$\textbf{if } \left(\frac{|Th_j(s_{k-1}) - Th_j(s_{k-2})|}{MAX(Th_j(s_{k-1}), Th_j(s_{k-2}))} \leq R_{th}\right) \textbf{ then}$$

$$T_j(s_k) = (T_j(s_{k-1}) * 2 < T_{max})?\ T_j(s_{k-1}) * 2 : T_{max}$$

$$\textbf{else}$$

$$T_j(s_k) = (T_j(s_{k-1})/2 > T_{min})?\ T_j(s_{k-1})/2 : T_{min}$$

$$\textbf{End if}$$

$$T(s_k) = MIN_{j=\{1..l\}} T_j(s_k)$$

Figure 2- Session duration estimation algorithm

This section illustrates a basic scenario that illustrates how the proposed model works. Suppose that a client *C* requests to download a content replicated in two servers S_1 and S_2 (n=2). In this scenario, the client is assumed to have two network interfaces/connections c_1 and c_2 (l=2) that can be used to communicate with the servers. Based on a certain best server selection mechanism, it is assumed that S_1 is the best server for C through c_1 and S_2 is the best server reachable through c_2.

- *First session:* In the startup session, *C* downloads respectively 65 Kilobytes from S_1 and S_2 through c_1 and c_2.
- *Intermediate sessions*: During the control phase preceding each session and based on the passive measurement of the average TCP throughput in the previous session, *C* calculates the chunk size for each interface, taking into account the session duration as presented in Equation 1. Then, *C* opens two TCP connections with S_1 and S_2 respectively and sends them first the next range of bytes to be downloaded through their associated interfaces.
- *Last session*: When the remaining amount of bytes to be transferred is less than the summation of the estimated chunks (Equation 2), the last download session is reached. In this session, the chunk to be downloaded from each network interface is calculated based on Equation 3.

4. Confidential Content Delivery

The confidentiality of the content to be downloaded can be of high importance for certain application domains. The security of the proposed scheme relies on two aspects: scrambling and encryption. The content of the whole requested file is shuffled before transmission and each chunk is encrypted using a secret key shared between the server and the client application.

4.1 Assumptions

It is assumed that users wishing to use the secure file transmission, must be authenticated first using a password based scheme. Then, a master secret key (MSK) is exchanged between the client and the server applications, using a password- authenticated key exchange mechanism such as [6] [2] [7] [8]. The analysis and choice of the key exchange mechanism is out of the scope of this paper. It is important to mention that the MSK is related to one content download and thus, should be renewed upon any new download.

4.2 Scrambling & Encryption

The client application generates two random numbers r_1(seed) and r_2 (salt). Then using a key derivation function (KDF) (e.g., [16], [28]), it generates a key k_0 = KDF (MSK, r_2) and encrypts r_1 using k_0. The integrity of k_0 is ensured using a keyed-hashed Message Authentication Code function (HMAC) having k_0 as an input and as a key at the same time. The client application then sends ($r_2 \parallel E_{k0}(r_1) \parallel HMAC_{k0}(k_0)$) to all the servers during the initialization session (preceding the startup session).

A server receiving ($r_2 \parallel E_{k0}(r_1) \parallel HMAC_{k0}(k_0)$) first calculates k_0 using KDF, MSK and r_2. It verifies the integrity of k_0 using the HMAC function and if the integrity check succeeds, it decrypts r_1 using k_0. After correctly obtaining k_0, r_1 and r_2, the server notifies the client application about successful initialization. The server then shuffles the content/file using a secure permutation (e.g., [5] and its

variants, [23]) based on r_1. Then it generates a key $k_1 = KDF(k_0, r_2)$, which is used to encrypt the file chunks. The encryption can be through a "pure" stream cipher (e.g., [27]) or a block cipher turned into a stream cipher using different modes of operations such as Cipher Feedback (CFB), Output Feedback (OFB) or Counter (CTR) modes [9]. The client application receives encrypted chunks. Knowing KDF, k_0 and r_2, it calculates k_1 and uses it to decrypt the encrypted chunks received in each session. After receiving all the file chunks, the client application has to de-shuffle the whole file by reversing the shuffling algorithm using the same seed (r_1).

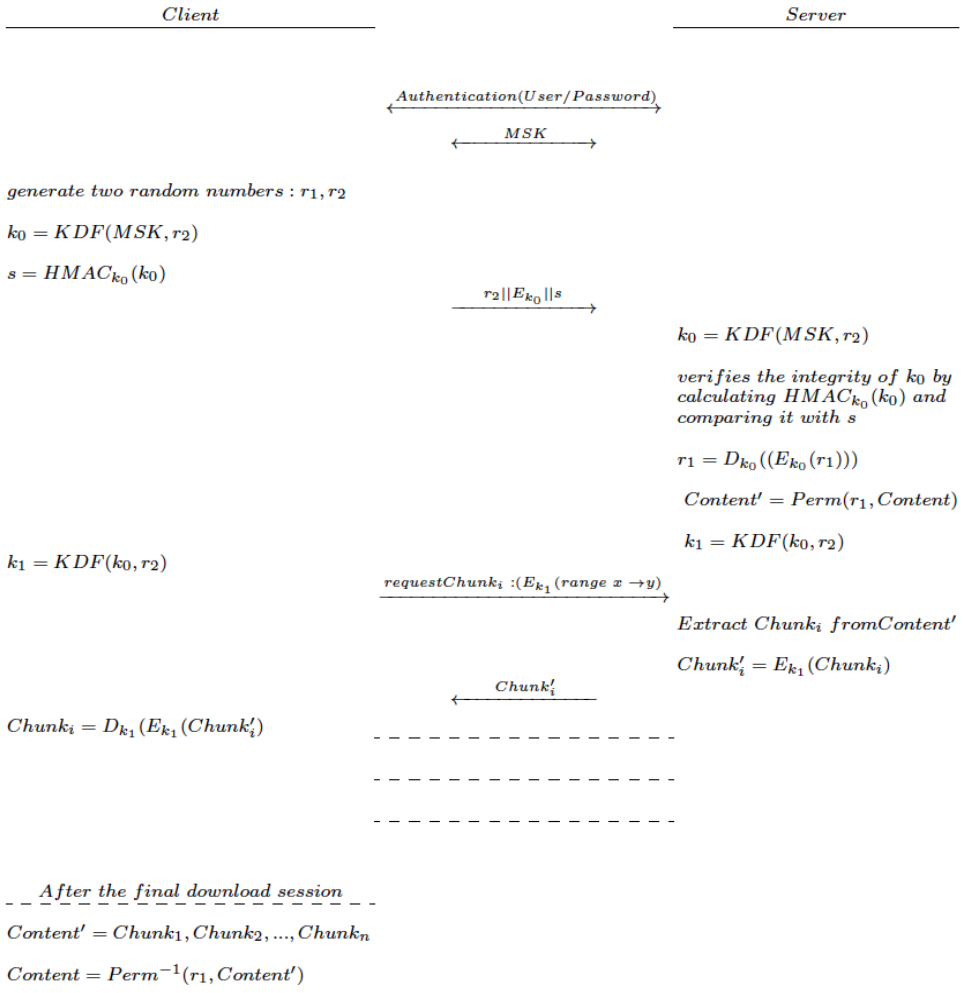


Figure 3- Key Exchange Mechanism

Additional security can be integrated by encrypting the messages sent from the client to the servers (range of bytes to be downloaded in each session) using a session key $k_x = KDF(k_{x-1}, k_{x-2})$, x=2...f, derived every session at both client and servers side. This will, as mentioned, improve the security of the scheme but increases the overhead of the scheme.

Figure 3illustrates the message exchanges required between a client and a server. For clarity purposes, only one server is considered; however, the same exchange can be generalized to the exchanges between a client and multiple servers. The chosen cryptographic algorithms in the proposed solution are:

- Encryption algorithm: Advanced Encryption Standard with key size of 256 bits in Counter mode (CTR) mode.
- HMAC function: HMAC with Secure Hash Algorithm SHA-256 [22]
- HMAC-based extract-and-expand key derivation function (HKDF) [16] as a key derivation function using the HMAC function selected above.

It is important to mention that the above-mentioned choices were taken based on the literature review of the existing cryptographic algorithms and the recommendation or approval of the standardization institutes such as NIST. However, the security scheme can be implemented using any other corresponding algorithm while taking into consideration the key size and algorithm input details. The proposed scheme relies on the security of the MSK (reference to the Kirchhoff's

principle[3]). The high complexity of breaking the scheme and thus "illegally" obtaining the content is based on the following points:

- The eavesdropper must be able listen to all the network paths connecting the client' interfaces to the servers.
- The eavesdropper needs to decrypt all the chunks;
- The eavesdropper should find the seed for de-shuffling the content.
- Decrypting parts of the content will not be useful to get any information since the content is scrambled but the complete operation must be done successfully to acquire information.

5. Model Validation

The performance enhancement of the proposed model is validated through the comparison with the classical point-to-point content delivery and with the point-to-multipoint classical content delivery (with fixed size chunks). In order to avoid the dynamicity of the Internet that makes the comparison between the different mechanisms and protocols irrelevant, the scenarios were built and run in an emulation testbed. This allows to regenerate the same configuration settings for testing and comparing the different protocols by running them in a sequential manner without any assumption related to the variation rate of network performance metrics.

The testbed (Figure 4) consists of: (i) one multi-homed client plugged simultaneously to two WiMAX network connections and (ii) two servers where each server is connected to one of the two WiMAX networks. All of these equipment is dedicated to the testbed and thus there is no external background traffic circulating in the WiMAX networks aside the traffic of the scenarios. The experimentations can be split into two measurement sets:

- the first measurement set is to choose the model' control parameters (T_{min}, T_{max}, and R_{th}).
- The results of the first measurement set (parameters values) are used in the second set to evaluate the overall performance of the model in comparison with other solutions.

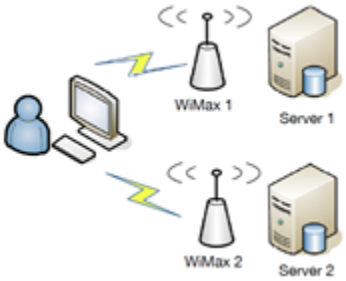


Figure 4- Validation Testbed

5.1 Control Parameters

In order to study the impact of of control parameters on the model performance and overhead, 270 scenarios were built and run: Each scenario consists in downloading a 10 GB file using fixed network connection speed (24Mbps download and 1.1Mbps upload) using 30 different combination values of protocol control parameters (Figure 5). Then, each scenario is repeated three times using three different seed values for generating the random background traffic. Thereafter, the same three seed values are used when changing the combination of protocol control parameters for obtaining comparable scenarios of experimentation. Control parameters vary as follows: T_{min} varies between 1 and 10 minutes, T_{max} varies between 20 and 30 minutes and R_{th} between 0.1 and 1. The variation of the parameters is done by changing the value of one parameter as indicated above while fixing the value of the two other parameters to the minimum value (1 minute for T_{min}, 20 minutes for T_{max} and 0.1 for R_{th}). For each measurement, the following latency values are calculated:

- T_m: the latency value measured when applying the proposed model. In this case, the content is delivered concurrently from the two servers through the two network connections according to the model presented in the previous section.
- T_1: the latency value measured when downloading the content in point-to-point mode from the first server.
- T_2: the latency value measured when downloading the content in point-to-point mode from the second server.

Figures 6, 7 and 8 illustrate the average enhancement (E) (Equation 5) brought by the proposed protocol vis-à-vis the point-to-point download scheme for different values of T_{min}, T_{max} and R_{th} respectively.

$$E = \frac{E_1 + E_2}{2} = \frac{\left(\frac{T_1 - T_m}{T_1}\right) + \left(\frac{T_2 - T_m}{T_2}\right)}{2} \quad (5)$$

The results show that better performance could be achieved using lower values of T_{min}, T_{max} and R_{th}. However, lower values require higher number of sessions and thus higher protocol overhead (e.g. processing). Thus, a tradeoff should be taken into account when tuning the values of these parameters.

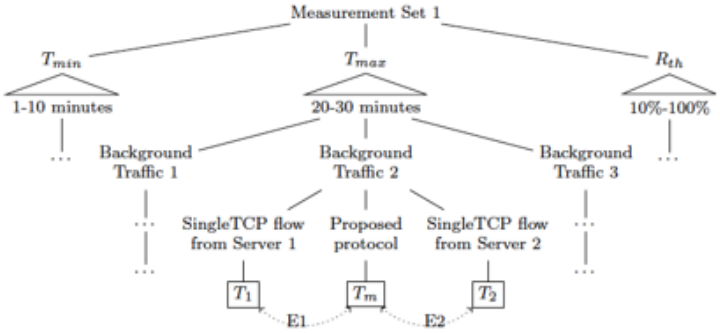
5.2 Model Evaluation

The second measurement set presented in Figure 9 aims at evaluating the proposed model based on the results obtained in the first measurement set.

In this set, external background traffic is used. Each experimentation scenario is defined by the following parameters: the network speed of the two clients' connections, the background traffic, and the size of the file to download. Each scenario is repeated four times by downloading the file:

1. simultaneously from the two servers according to the proposed model.
2. from the first server in classical point-to-point mode through the first network connection.
3. from the second server in classical point-to-point mode through the second network connection.
4. simultaneously from the two servers associated to the network interfaces but in one session where the size of the chunk to be downloaded from each server is fixed and estimated proportionally to the network interface connection speed.

3 A cryptographic system should be secure even if everything about the system, except the key, is public knowledge



- *File Size= 10GB*
- *DownSpeed1=24Mbps*
- *DownSpeed2=8Mbps*
- *E_1: the enhancement brought by the proposed protocol vis-à-vis the point-to-point download from server 1.*
- *E_2: the enhancement brought by the proposed protocol vis-à-vis the point-to-point download from server 2.*

Figure 5- Measurement Set 1

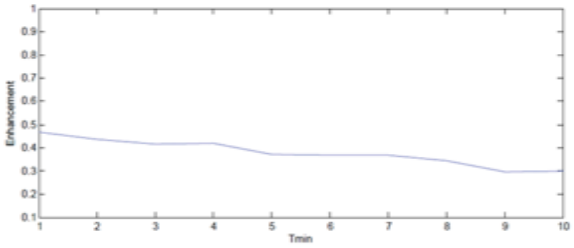


Figure 6- Average Enhancement vs. T_{min}

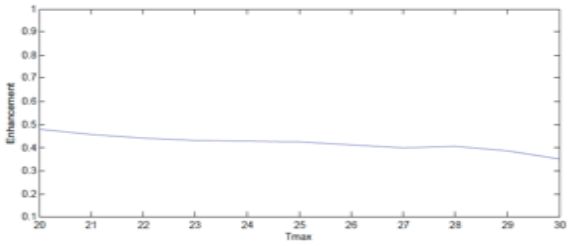


Figure 7- Average Enhancement vs. T_{max}

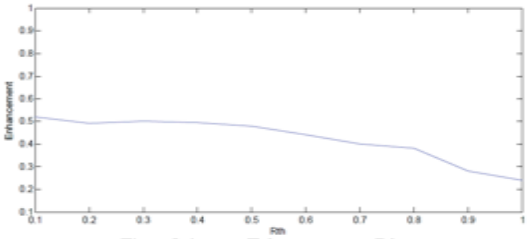


Figure 9-Average Enhancement vs. *Rth*

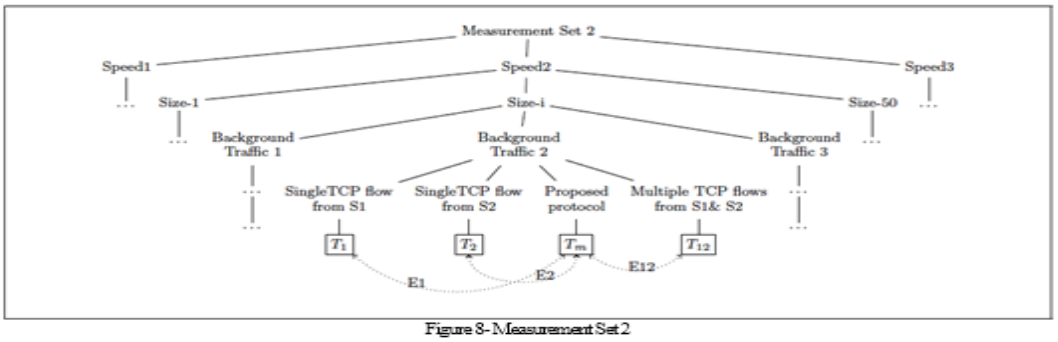


Figure 8- Measurement Set 2

Table 2 - Scenario parameters

Parameter	*Values*
Connection Speeds	Download: 8 Mbps, Upload: 1.1 Mbps Download: 12 Mbps, Upload: 1.1 Mbps Download: 24 Mbps, Upload: 1.1 Mbps
File Size	50 Different sizes between 10MB and 10GB
Background Traffic	Random Number of TCP connections (1-100) Random file size per each TCP connection (10MB-10GB)

Table 2 illustrates the value ranges of the different parameters in the emulated scenarios. To repeat the same scenario, the same seed and salt were used for generating the same random numbers. Besides, the following values of protocol control parameters were considered: 1 minute for T_{min}, 20 minutes for T_{max} and 0.1 for R_{th}. As mentioned previously (in section 3.1), such low values require more processing. However, the purpose of this choice is to better evaluate the performance of the proposed model (the impact of tuning these parameters has been addressed in our experimentations but could not be included in this paper due to space limitation). The results collected from such emulated experiments are assumed to be analogous to the ones collected from downloading files of different sizes from different servers distributed in the Internet and in different times. During a three-month period of emulations, the results of 450 scenarios were collected in the following way: For each combination of network connection speed among the three, the downloads of 50 files having different sizes ranged from 10MB and 10GB were tested. Each download is repeated three times by generating different random background traffic. Then, each of the 450 scenarios is repeated four times with the same settings (for being comparable) to measure the three latency values previously mentioned (T_1, T_2 and T_m) in addition to T_{12} which is the latency value measured when downloading the file from the two servers simultaneously in one session. The size of the two chunks to be downloaded from the two servers is proportional to the speed of their network connections. In order to compare the proposed content delivery model with the point-to-point one, the enhancement (E) (Equation 5) is calculated. Figure 10 illustrates the Cumulative Distributed Function (CDF) of the metric enhancement estimated from the performed scenarios when compared to classical point-to-point download (single flow) and to multiple-servers download scheme (Multiple flows). The comparison with single flow case shows that more than 80% of the scenarios were subject to an enhancement between 45% and 80%. The rest of scenarios still witness an enhancement between 18% and 45%. The expected enhancement[4] is near 55%. The comparison between T_m and T_{12} (multiple flows case) reflects the enhancement (Equation 6) provided by the proposed model which divides the download into sessions having adaptive durations. It is obvious that the enhancement is not the same as the classical point-to-point download but still there are 60% of the scenarios that improved for 10% to 55%. The Expected enhancement value is thus around 16%.

$$E_{12} = \left(\frac{T_{12} - T_m}{T_{12}}\right) \qquad (6)$$

Finally, the performance of the proposed model has been measured vis-à-vis the size of file to be downloaded. It is clearly seen in Figure 11[5] that in general the higher the file size is, the better the performance of the model is. However, the average enhancement is noticeable despite of the file size (ranging between 40% and 60%). The fact that the proposed model could provide better results for larger files is that larger files require higher number of sessions. This will allow the model to converge more rapidly to estimate more accurately the average throughput (passive measurement) and thus to have more accurate estimation of session duration. This provides the model with more precise adaptivity to the dynamicity of network performance, which allows pooling the network capacity from the two servers (of the scenario) in a more efficient way and thus results in the latency enhancement. The proposed application-level protocol can be classified as "smart application". It is smart in the sense that it automatically improves the latency based on the characteristics at the physical layer, link layer, and transport layer which are taken into consideration along the running time of the application. Such cross-layer awareness is illustrated by the following dependencies: (1) Each client's network interface is associated to a specific network interface of the best server (best server selection procedure). (2) The network throughput is measured passively to estimate the current session duration. (3) The chunk size of each connection is calculated based on the estimated session duration. (4) Each TCP stream is communicated through the associated network interface.

6. Conclusion

In this paper, we propose an adaptive session duration model to improve the con- tent distribution in overlay networks. Our model takes advantage from the con- tent replication and multi-homing facilities which are widely available nowadays.

Then, it consists of pooling the network capacity in space and time by striping data across multiple TCP sockets (i.e., associated to the network interfaces) that simultaneously download the content chunks from their associated best servers. This is achieved in multiple sessions having adaptive durations estimated based on the passive measurement of network performance. The proposed solution was enhanced with encryption and key exchange schemes in order to ensure the confidentiality of the content download. Our extensive measurements clearly show how our solution outperforms the existing content delivery techniques by considerably decreasing the latency. Our traces show that with high probability, the delivery time could be decreased to an amount smaller than half of its value when applying our model for content delivery instead of the classic point-to-point content delivery method. It also outperforms the classic concurrent download method by an expected value of 16%. This is due to the fact that it combines the bandwidth aggregation facility in multi-homing environment with our adaptive session duration mechanism.

4 the average value over all the traces.

5 the range of the file size is from 10 MB to 10 GB. The log scale is used to draw this wide range of values. The enhancement is calculated in comparison with the point-to-point case.

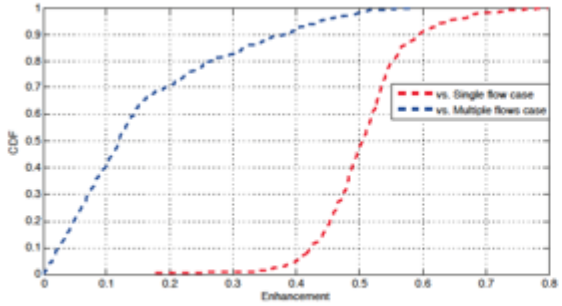


Figure 10-Average Enhancement

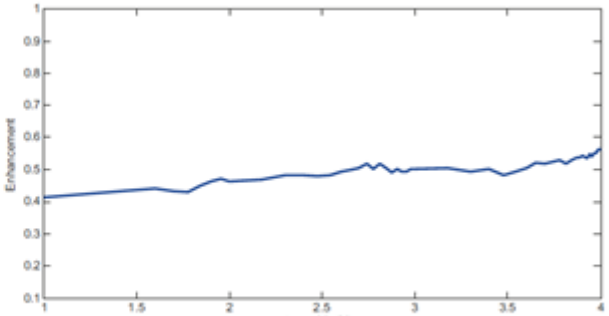


Figure 11-Enhancement per file size

References

[1] Akamai Technologies, Inc. Http://www.akamai.com/ , Last Accessed on 16/02/2016

[2] Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: A. Menezes (ed.) Topics in Cryptology CT-RSA 2005, Lecture Notes in Computer Science, vol. 3376, pp. 191–208. Springer Berlin Heidelberg (2005).

[3] Adhikari, V., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., Zhang, Z.L.: Unreeling netflix: Understanding and improving multi-CDN movie delivery. In: INFOCOM, 2012, Proceedings IEEE, pp. 1620–1628 (2012)

[4] Alzoubi, H.A., Lee, S., Rabinovich, M., Spatscheck, O., van der Merwe, J.E.: A practical architecture for an anycast cdn. ACM Transactions on the Web 5(4), 17 (2011)

[5] Black, P.E.: Fisher-yates shuffle. Dictionary of Algorithms and Data Structures 19 (2005)

[6] Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: B. Preneel (ed.) Advances in Cryptology EUROCRYPT 2000, Lecture Notes in Computer Science, vol. 1807, pp. 156–171. Springer Berlin Heidelberg (2000)

[7] Brusilovsky, A., Faynberg, I., Zeltsan, Z., Patel, S.: Password-Authenticated Key (PAK) Diffie-Hellman Exchange. RFC 5683 (Informational) (2010). URL http://www.ietf.org/ rfc/rfc5683.txt

[8] Chien, H.y.: Provably secure authenticated diffie-hellman key exchange for resource-limited smart card. Journal of Shanghai Jiaotong University (Science) 19(4), 436–439 (2014). DOI 10.1007/s12204-014-1521-7. URL http://dx.doi.org/10.1007/s12204-014-1521-7, Last Accessed on 16/02/2016

[9] Dworkin, M.: Recommendation for block cipher modes of operation: Methods and techniques. Special Publication 800-38A, National Institute of Standards and Technology (2001). URL http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf, Last Accessed on 16/02/2016

[10] Ford A. Raiciu C., H.M., O., B.: TCP extensions for multipath operation with multiple addresses. IETF Internet Draft (2014)

[11] G. Bertrand E. Stephan, G.W.T.B.P.E., Ma, K.: Use cases for cdni. IETF Draft (2012)

[12] Haddow, T., Ho, S.W., Ledlie, J., Lumezanu, C., Draief, M., Pietzuch, P.: On the feasibility of bandwidth detouring. In: Proceedings of the 12th International Conference on Passive and Active Measurement, PAM'11, pp. 81–91. Springer-Verlag, Berlin, Heidelberg (2011)

[13] Hsieh, H.Y., Sivakumar, R.: pTCP: An end-to-end transport layer protocol for striped connections. In: Proceedings of the 10th IEEE International Conference on Network Protocols, ICNP '02, pp. 24–33. IEEE Computer Society, Washington, DC, USA (2002)

[14] Hsieh, H.Y., Sivakumar, R.: A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. Wireless Networks 11(1-2), 99–114 (2005)

[15] Hu, N., 0002, L.L., Mao, Z.M., Steenkiste, P., Wang, J.: A measurement study of internet bottlenecks. In: INFOCOM, pp. 1689–1700. IEEE (2005)

[16] Krawczyk, H., Eronen, P.: RFC 5869: HMAC-based extract-and-expand key derivation function (HKDF). IETF RFC (2010). https://tools.ietf.org/html/rfc5869 , Last Accessed on 16/02/2016

[17] Labovitz, C., Ahuja, A., Bose, A., Jahanian, F.: Delayed internet routing convergence. IEEE/ACM Trans. Netw. 9(3), 293–306 (2001)

[18] Liu, H.H., Wang, Y., Yang, Y.R., Wang, H., Tian, C.: Optimizing cost and performance for content multihoming. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM '12, pp. 371–382. ACM, New York, NY, USA (2012)

[19] Logg, C., Cottrell, L., Navratil, J.: Experiences in traceroute and available bandwidth change analysis. In: Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality, Net '04, pp. 247–252. ACM, New York, NY, USA (2004)

[20] Malli, M.: Fast distribution of replicated content to multi-homed clients. ACEEE International Journal of Information Technology 3(2), 7 (2013)

[21] Malli, M., Barakat, C., Dabbous, W.: CHESS: An application-aware space for enhanced scalable services in overlay networks. Computer Communications 31(6), 1239–1253 (2008)

[22] NIST: Secure hash standard (shs) (2012). URL http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf , Last Accessed on 16/02/2016
[23] Ohrimenko, O., Goodrich, M.T., Tamassia, R., Upfal, E.: The melbourne shuffle: Improving oblivious storage in the cloud. In: Automata, Languages, and Programming, pp. 556–567. Springer (2014)
[24] Peterson, R.S.: Efficient content distribution with managed swarms. Ph.D. thesis, Ithaca, NY, USA (2012). AAI3506806
[25] Peterson, R.S., Wong, B., Sirer, E.G.: A content propagation metric for efficient content distribution. SIGCOMM Computer Communication Review 41(4), 326–337 (2011)
[26] Poese, I., Frank, B., Ager, B., Smaragdakis, G., Feldmann, A.: Improving content delivery using provider-aided distance information. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, pp. 22–34. ACM, New York, NY, USA (2010)
[27] Rivest, R.L., Schuldt, J.C.N.: Spritz-a spongy rc4-like stream cipher and hash function. Presented at Charles River Crypto Day (2014-10-24) (2014). URL http://bostoncryptoday.wordpress.com/2014/09/07/friday-october-24-2014-at-mit/ , Last Accessed on 16/02/2016
[28] Turan, M.S., Barker, E., Burr, W., Chen, L.: Recommendation for password-based key derivation, part 1: Storage applications. NIST SP 800-132 (2010). URL http://csrc.nist.gov/publications/nistpubs/800132/nist-sp800-132.pdf
[29] Wischik, D., Handley, M., Braun, M.B.: The resource pooling principle. SIGCOMM Computer Communication Review 38(5), 47–52 (2008)
[30] Wischik, D., Handley, M., Raiciu, C.: Control of multipath TCP and optimization of multipath routing in the internet. In: Proceedings of the 3rd Euro-NF Conference on Network Control and Optimization, NET-COOP '09, pp. 204–218. Springer-Verlag, Berlin, Heidelberg (2009)
[31] Zhang, M., Lai, J., Krishnamurthy, A., Peterson, L., Wang, R.: 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, ATEC '04, pp. 8–8. USENIX Association, Berkeley, CA, USA (2004)

Mohammad G. Malli obtained the Diploma in Electrical and Electronics Engineering from the Lebanese University (Lebanon). He received the Master and PhD Degrees in Networking and Distributed Systems from the University of Nice Sophia Antipolis (France). He is currently the coordinator of the ITC program in the Arab Open University - Lebanon. His research interests lie in the areas of Open Learning, Mobile Computing, Computer Networking, and Social Networking.

Hassan Sbeity received the Dipl.-Ing degree in Electrical Engineering and Information Science form the Ruhr Universitaet Bochum (Germany) in 1993. He worked many years in the development of PC I/O devices and their drivers at EBS in Germany. In 2001, he received the DEA (Masters) in Informatics, Modeling and Intensive Calculation from AUF and the Lebanese University in Beirut. In 2005, He received PhD in Informatics at "Valenciennes et du Hainaut Cambresis", LAMIH ROI (France) with collaboration of the University of Ghent (ELIS), Belgium. He is currently Assistant professor at the Arab Open University Lebanon branch (since 2003). His main research interests include open learning, computer architecture, application parallelization, Mobile computing, Multimedia applications, and memory optimization of embedded systems.

Ahmad Fadlallah received his Engineering Diploma in Electrical Engineering and Electronics from the Lebanese University (Lebanon) in 2001. In 2003, he received the DEA (MSc.) in Telecommunication Networks from the Lebanese University and University of Saint Joseph-Lebanon. In 2008, he received a PhD in Computer science and Networking from Telecom ParisTech-France. He is currently Assistant professor at the Information Technology and Computing Department in the Arab Open University – Lebanon branch (since 2008). His research interests lie in the areas of open and e-learning, mobile networks, multimedia services and computer & network security.

Ali Hodroj earned his Computer Communication Engineering degree from the Lebanese International University in 2012. Then, he has obtained his Master degree in networking from the Lebanese University in 2013. In September 2015, he has started his PhD thesis entitled "Enhancing content delivery in multi-homed broadband mobile networks" under the joint guardianship of Saint Joseph University and Rennes University.

Abed Ellatif Samhat received an engineering diploma from the Lebanese University, Beirut, in 2000 and PhD in computer science from the Pierre et Marie Curie University, Paris, in 2004. He was with Orange Labs-Paris from 2005 to 2008 where he has been involved in several national and European projects including Ambient Networks and Gandalf. In 2009, he joined Lebanese University, Beirut and he is currently a professor at the Faculty of Engineering. His areas of interest include heterogeneous wireless networks, access selection and mobility management.