future internet

Article

Joint Scalable Video Coding and Transcoding Solutions for Fog-Computing-Assisted DASH Video Applications

Majd Nafeh [1], Arash Bozorgchenani [2] and Daniele Tarchi [1,*]

[1] Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40121 Bologna, Italy
[2] School of Computing and Communications, Lancaster University, Lancaster LA1 4YQ, UK
* Correspondence: daniele.tarchi@unibo.it

Abstract: Video streaming solutions have increased their importance in the last decade, enabling video on demand (VoD) services. Among several innovative services, 5G and Beyond 5G (B5G) systems consider the possibility of providing VoD-based solutions for surveillance applications, citizen information and e-tourism applications, to name a few. Although the majority of the implemented solutions resort to a centralized cloud-based approach, the interest in edge/fog-based approaches is increasing. Fog-based VoD services result in fulfilling the stringent low-latency requirement of 5G and B5G networks. In the following, by resorting to the Dynamic Adaptive Streaming over HTTP (DASH) technique, we design a video-segment deployment algorithm for streaming services in a fog computing environment. In particular, by exploiting the inherent adaptation of the DASH approach, we embed in the system a joint transcoding and scalable video coding (SVC) approach able to deploy at run-time the video segments upon the user's request. With this in mind, two algorithms have been developed aiming at maximizing the marginal gain with respect to a pre-defined delay threshold and enabling video quality downgrade for faster video deployment. Numerical results demonstrate that by effectively mapping the video segments, it is possible to minimize the streaming latency while maximising the users' target video quality.

Keywords: fog computing; DASH; scalable video coding; transcoding

Citation: Nafeh, M.; Bozorgchenani, A.; Tarchi, D. Joint Scalable Video Coding and Transcoding Solutions for Fog-Computing-Assisted DASH Video Applications. *Future Internet* **2022**, *14*, 268. https://doi.org/10.3390/fi14090268

Academic Editor: Gyu Myoung Lee

Received: 25 July 2022
Accepted: 15 September 2022
Published: 17 September 2022

1. Introduction

Multimedia streaming has been on the hype during the last decade, enabling a smarter video-watching system. It is clear to all how well-known brands, e.g., YouTube, Netflix, Prime Video, Hulu, Disney+, etc., have completely changed the way people watch video content. According to the statistics, the majority of Internet traffic originates from video streaming applications such as Netflix, YouTube, etc. [1]. This spread is heavily supported by a set of new technologies introduced in the last decades that enabled a smart implementation of streaming solutions for everyone. Among others, Dynamic Adaptive Streaming over HTTP (DASH) made it possible to stream large video content while adapting to the variable network conditions and exploiting HTTP protocol [2], hence, overcoming older protocols (e.g., Real-time Transport Protocol—RTP) suffering from reduced client compatibility and gateway port filtering.

In order to implement a proper video streaming application, new network protocols required a huge amount of storage space and computing efforts. On the one hand, videos require lots of storage space. Despite the recently introduced more efficient encoding standards, video content still remains one of the heaviest in terms of data size [3]. At the same time, an increased number of devices, video codecs, and user requirements have pushed towards scalable approaches where the same video content should be adapted to the users' requests [4]. In order to do this, additional processing capabilities are required. Therefore, cloud computing facilities soon became one of the main enabling technologies for video streaming services [5].

Despite big efforts in integrating cloud computing in streaming scenarios, the last years have been characterized by an impressive increase in multimedia content, highlighting some disadvantages inherently present in the centralized cloud computing approach, mainly in terms of high latency and scalability issues. In future 5G mobile networks, the low end-to-end communication latency and high available bandwidth allows the mobile users to stream with high quality. Fog [6,7] and edge [8,9] computing paradigms, by bringing the services closer to the users, have indeed increased the interest of the multimedia world as a feasible way of solving latency and scalability, which leads towards satisfying the requirements of 5G networks. On one side, edge and fog computing nodes can be installed nearer to the users, reducing the latency, and, on the other side, their numbers can be increased, reducing the problem of scalability [10–12].

By exploiting the fog computing paradigm, we are aiming at integrating the DASH approach into a fog computing scenario. In particular, by exploiting the DASH media segment organization, we propose a joint storage and processing approach that, leveraging on a fog computing infrastructure, allows to optimally place the video segments in order to maximize the user experience. In particular, the proposed approach allows taking into account the possible limitations of the fog nodes in terms of the amount of data that can be stored so as to build an implementable solution for delivering videos at the edge. In addition, based on the fact that videos are usually organized in multiple quality levels, targeted to different user requests, we aim at integrating an on-demand approach that allows to jointly consider the video quality scalability and transcoding. While scalable video coding (SVC) allows to interleave different streams, typically one basic stream and one additional stream, so as to increase the quality from the basic to the targeted higher quality, in the case of transcoding, the quality can be changed to fit the user requests [13]. To this aim, a proper procedure is proposed for integrating both scalability and transcoding in the fog computing architecture, aiming at minimizing the overall video delivery latency. This can be made possible in DASH, where the video content management is based on an information collecting procedure, and the video content is organized through a proper media presentation description (MPD) format [2]. MPD includes segment information such as video resolution and bit rates, used as metrics for the quality of experience (QoE) evaluation. Based on the target quality level, a decision is made with possible transcoding or scalable video coding.

Despite some recent research in this field, the possibility of exploiting fog and edge nodes for assisting DASH-based video delivery is still unexplored in depth. Our goal is to focus on two algorithms able to optimize the segment placement in a distributed fog network. To this aim, both communication and storage constraints have been considered, aiming at optimally placing the video segments while respecting both target quality and latency requirements. While the first proposed algorithm considers the target quality as a primary constraint, in the second algorithm, the quality could be degraded in case the latency requirement cannot be respected. In particular, both SVC (based on a cloud repository) and transcoding (based on an edge server) techniques are considered when evaluating the latency and the quality requirements.

The paper is structured in this way. In Section 2, the literature background in the area of VoD and its implementation through edge and fog computing infrastructure is reviewed. In Section 3, the system model is described with a particular focus on the latency model, while in Section 4, the problem is formulated by introducing a cost function aiming to map the considered scenario. The proposed algorithms are explained in Section 5, while in Section 6, the numerical results obtained through computer simulations are shown. Finally, conclusions are drawn in Section 7.

2. Literature Review

Multimedia is considered one of the major application areas for 5G and B5G services [14]. As an example, video streaming and video-on-demand applications are considered as one of the pillars for smart city services [15]. To this aim, some papers developed specific solutions for including videos in different smart city applications. As an example, in [16], the authors proposed a smart city framework for video surveillance, where multiple cameras disseminated within a city are exploited. This proposed framework, named UTOPIA, allows to perform a video analysis exploiting MapReduce. The authors propose to use the developed solution for video surveillance applications. The idea of using videos for implementing an intelligent transportation system is instead proposed in [17]. The authors propose to use road-side cameras jointly with road traffic monitoring servers. The visual IoT idea is instead introduced in [18]. Trying to merge multimedia content and IoT, the authors envisaged modelling the cameras as sensors for implementing smart city services. The authors first performed an analysis of the requirements of the proposed visual IoT approach, and then an architecture enabling the visual IoT concept to operate was introduced. Additionally, in [19], smart surveillance applications based on IoT sensors are considered. In this context, a heterogeneous network scenario is considered to implement a multi-purpose real-time video surveillance application to be applied to both smart cities and border control. The proposed solution considers three main aspects, which are the QoS requirements of the real-time video application, the cost-benefit of the spectrum allocation and the time constraints involved in vertical handover operations.

On the other side, MEC solutions in 5G scenarios have been considered in several papers also enabling multimedia services [20]. As an example, in [21], the authors explored the possibility of using an edge computing facility for content distribution in a smart city environment. Through the MEC, the system is capable of offering data storage and processing capabilities closer to data sources and data consumers. The authors explored how MEC can be used when user mobility and resource allocation should be managed. The possibility of using DASH in the cloud architecture has been examined in [22]. The authors proposed to use three different cloud computing companies operating in nine different data centres for streaming a video. The conclusion was that deploying DASH in the cloud is useful and that the performance of DASH is enhanced when using the cloud; this is even more true when the data centre is near the users. Another paper investigated the use of the cloud for the transcoding function [23]. The paper proposed a scheduling algorithm that optimizes the video transcoding time (VTT), i.e., the time needed to transcode the video, which in turn influences the video transcoding mode (VTM), i.e., the process that overlooks the transcoding requests. It is demonstrated that the algorithm allows a faster transcoding process, better management of tasks and a continuous stream. The authors in [24] propose a partial transcoding method for content management where each content is encoded into different bitrates and split into segments. They aimed at minimizing the cost, which includes storage and computing costs. In [25], the analytic hierarchy process (AHP) method was proposed. In this paper, a multi-tier fog architecture iwas used and several metrics such as bandwidth, latency and financial costs are taken into consideration. Based on all the collected data and the AHP output, the most suitable fog nodes are used for the stream. This has shown to be an advantage from the QoE side as well on the delay side since it is reduced remarkably. Additionally, it is shown that the bandwidth is increased and the cost is minimized. A similar paper has looked into optimizing the number of fog nodes needed to reduce the delay [26]. This paper uses the distance between the fog nodes and the gateways. It measures the latency by leveraging the k-means clustering method in different ways, calculating the distance between the fog nodes and the gateways with distinct approaches that consider the number of fog nodes and the number of gateways. It proved that there are several ways to execute the placement of fog nodes, and all of them lead to much lower latency. Ref, [27] instead considered the creation of a fog network by the means of an online optimization that is capable of improving the functions of the network. The pilot fog node shared the workload with the nearby fog nodes to offload tasks. The online optimization algorithm observed the neighbouring fog nodes and picked the appropriate ones based on their capabilities to form the network. An offload optimization algorithm is used to disperse tasks. Following those methods, the maximum delay in the network is effectively cut down and the task offloading is enhanced. Another paper explored the relocation of the transcoding node closer to the users [28]. The transcoding node would be aware of the user's environment and would receive the media requested. Since the node knows the media solicitor, it can transcode the segments properly for the user; this leads to a system that is flexible when it comes to contrasting users, more efficient with fewer segment levels and more scalable as it can handle more users.

3. System Model

Let us consider having a set $\mathcal{U}$ composed of N users and a set $\mathcal{V}$ composed of M videos available for their requests. The deployed network is composed of a set $\mathcal{F}$ of K fog nodes; in addition, one remote cloud facility is considered. Each user is able to request one video with a certain quality level, so that each request can be mapped to a given target quality level, identifying the user request QoE. We consider that there are a limited number of quality levels corresponding to Q where $q = 1$ corresponds to the worst quality and $q = Q$ corresponds to the highest quality.

Let us define $\rho_n = \langle m, q \rangle$ as the tuple identifying the requested mth video with the q-level quality by the nth user. By leveraging on the distributed fog nodes facility, each video is supposed to be segmented, where the mth video is segmented in P_m segments whose size can be fixed or change depending on the considered encoder and video characteristics [29]. Moreover, transcoding and scalability functions are foreseen to be used in the deployed scenario. While transcoding allows to receive a given video segment with a target quality lower than that of the video already deployed, the scalability allows to request an enhanced video stream able to cope with the missing information for achieving higher quality with respect to the quality of the stored video.

We can consider an allocation matrix $A(k, p_m)$ as a function allowing us to map the presence of the p_m-th segment belonging to the m-th video on the k-th fog node. Since each segment can be stored with different quality levels, it is possible to define the allocation matrix as:

$$A(k, p_m) = \begin{cases} q & \text{if } C_1 \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where C_1 is the condition stating that the p_m-th segment belonging to the m-th video is stored on the k-th fog node with quality level q. Since fog nodes are supposed to be storage-capacity constrained, a maximum amount of data can be stored in each node. Let us assume that the generic k-th node can buffer up to B_k bytes.

Based on the video request table $\mathbb{P}$, composed by n records, each one identifying the video and the quality requested by each user, each video segment will be mapped on a given fog node in order to minimize the overall delay for receiving the requested video from the user's side. To this aim, the cost of any deployment configuration can be modeled as the overall time needed for receiving all the segments of the requested video with the target quality level, i.e.:

$$T(n) = \sum_{p_m = 1_m}^{P_m} t^{k,n}(p_m) \tag{2}$$

corresponding to the time needed from the user n's side to receive all the video segments of the mth video, where:

$$t^{k,n}(p_m) = \begin{cases} \frac{d^{q}_{p_m}}{r_{k,T}} + \delta_{tc}(d^{q}_{p_m}, d^{\bar{q}_n}_{p_m}) + \frac{d^{\bar{q}_n}_{p_m}}{r_{T,n}} & \text{if } \bar{q}_n < q \wedge A(k, p_m) = q \\ \frac{d^{q}_{p_m}}{r_{k,n}} & \text{if } \bar{q}_n = q \wedge A(k, p_m) = q \\ \frac{d^{q}_{p_m}}{r_{k,n}} + \frac{d^{(\bar{q}_n - q)}_{p_m}}{r_{0,n}} & \text{if } \bar{q}_n > q \wedge A(k, p_m) = q \\ \frac{d^{\bar{q}_n}_{p_m}}{r_{0,n}} & \text{otherwise} \end{cases} \tag{3}$$

and $d_{p_m}^q$ is the size of the p_m-th segment of the m-th video with quality level q, $r_{i,j}$ is the data rate between node i and node j, $\bar{q}_n$ is the target quality of user n, and $\delta_{tc}(d_{p_m}^q, d_{p_m}^{\bar{q}_n})$ is the time requested for transcoding the segment $d_{p_m}^q$ to the target quality segment $d_{p_m}^{\bar{q}_n}$.

Figure 1 depicts the architecture of the proposed fog-assisted DASH scenario. We have considered having one transcoding node, identified with index T able to process multiple segments at the same time. This node can be identified with an edge computing facility collocated with the access point (AP) in the area. Moreover, a remote cloud facility is considered, as identified with index 0, where the up-scaling data are stored, identified as $d_{p_m}^{(\bar{q}_n - q)}$, needed for scaling up the quality from level $\bar{q}_n$ to level q. The data rate of the links is a function of the distance between node i and node j such as:

$$r_{i,j} = r \cdot \left(\frac{10}{d_{i,j}} \right)^2 \tag{4}$$

derived from the log-distance path loss model [30].

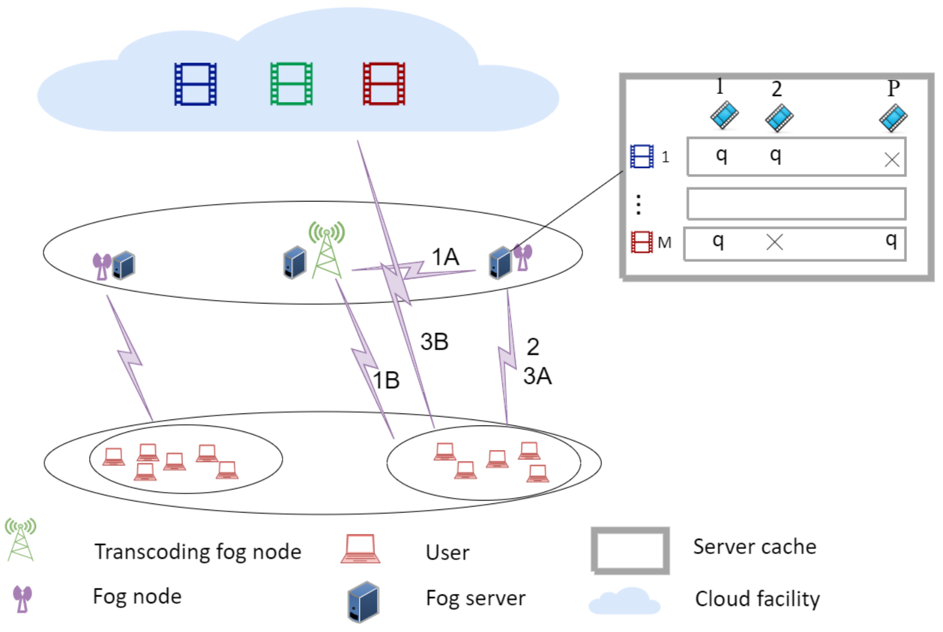


Figure 1. The Proposed Fog-assisted DASH Architecture.

Depending on the matching between the target quality level $\bar{q}_n$ and the quality level q of the p_m-th segment to be received, $t^{k,n}(p_m)$ can have three values:

- If the quality q is higher than the target quality $\bar{q}_n$, the download time corresponds to the latency between the source kth fog node to the transcoding node plus the transcoding time from quality q to quality $\bar{q}_n$ plus the latency from the transcoding node to the requesting user (case 1.A and 1.B in Figure 1).
- If the quality q of the stored video and the target quality matches the latency, the download time corresponds to the time between the k-th fog node and the requesting user (case 2 in Figure 1).
- If the quality q is lower than the target quality $\bar{q}_n$, the system allows to gain the enhancement from the cloud to target the requested quality level. This results in the latency from the kth node to the requesting user of the q level segment plus the latency of the missing part from the cloud to the requesting node (case 3.A and 3.B in Figure 1).

As an additional case, we consider when the segment is not present in any node. In that case, we suppose that it can be entirely downloaded from the cloud facility.

4. Problem Formulation

Streaming media consumes a lot of bandwidth; moreover, with the increased request of high-quality videos, their size becomes too heavy. Even if new compression standards have been introduced [31], that alone is not enough. DASH can help in delivering the most suitable quality to the user with reduced bandwidth wastage [32]. Fog computing would leverage computation and storage by distributing computing to various nodes, therefore reducing latency by having wider geographical distribution. The combination allows to break the video into segments and distribute them to fog nodes, which in turn makes it reach the user faster and with better quality.

By assuming that the qth quality level is mapped on an average video bit rate equal to $VBR(q)$, the video segments delivery is constrained to:

$$\sum_{p_m=1_m}^{\bar{p}_m} t^{k,n}(p_m) \leq \sum_{p_m=1_m}^{\bar{p}_m} \frac{d^{\bar{q}p_m}}{VBR(\bar{q})} \quad \forall \bar{p}_m = \{1_m, \ldots, P_m\} \tag{5}$$

in order to avoid video latency disruption during the streaming. Equation (5) models that the delivery time for the first segment should be lower than the average playout time, the delivery time of the first two segments should be lower than the playout time of both, and so on.

At the same time, the video quality, in terms of video bit rate, is a qualitative feature to be considered during the video streaming; thus, a reduced quality is sometimes tolerated, while a video stuck represented by underbuffering is undesired. To this aim, a relaxing assumption could be included, allowing to retrieve a given segment with a lower quality.

Given a certain initial video segment placement, the goal of the problem is to optimize the placement of the segments in order to minimize the delivery latency, defined as:

$$\mathbf{P1}: \ \min \sum_n T(n) \tag{6}$$

such that

$$\mathbf{C1}: \sum_{p_m=1_m}^{\bar{p}_m} t^{k,n}(p_m) \leq \sum_{p_m=1_m}^{\bar{p}_m} \frac{d^{\bar{q}p_m}}{VBR(\bar{q})} \qquad \forall \bar{p}_m = \{1_m, \ldots, P_m\} \tag{7a}$$

$$\mathbf{C2}: A(k, p_m) = \begin{cases} q & \text{if } C_1 \\ 0 & \text{otherwise} \end{cases} \tag{7b}$$

$$\mathbf{C3}: \mathcal{Q}\left(\sum_{m,p_m} A(k, p_m) \right) \leq B_k \forall k \tag{7c}$$

$$\mathbf{C4}: \sum_k A(k, p_m) < \sum_{m,p_m} A(k, p_m) \tag{7d}$$

$$\mathbf{C5}: A^0(k, p_m) = rand\{0, q\} \tag{7e}$$

where $\mathcal{Q}(\cdot)$ is a function able to map the quality level into a given amount of Bytes. Constraint (7d) posits that the amount of segments for all the videos that can be stored in the fog computing environments is smaller than all the segments composing all the videos. Indeed, the fog computing environment is a resource-limited scenario and is able to store only a portion of the videos. The remaining are supposed to be stored on the remote cloud. Constraint (7e) posits that the starting condition is that the segments are randomly scattered on the fog nodes with a random quality. It is worth noticing that despite (6) minimizing the overall delay, its value depends on the delay components, as defined in (2), which, in turn, depend on the allocation, following (3).

5. Distributed Mapping Algorithms

Since the problem outlined in (6) is NP-hard due to the presence of multiple segments to be deployed on multiple nodes while respecting the time constraints as defined in (3), we resort to two heuristic algorithms. In particular, we focus on two algorithms, named *marginal gain maximization* and *quality downgrade*. While in the first, we aim at minimizing the difference between the delivery deadline and the latency through a proper mapping, in the second algorithm, we aim at respecting the latency as a hard constraint while allowing a quality downgrade in selecting the proper segment.

The marginal gain maximization algorithm gains from the fact that for reducing the outage defined in (5), we can first of all assign the segments that have the highest marginal gain between the playback rate and the transmission rate, i.e.,

$$\max\left\{\frac{d^q_{p_m}}{VBR(q)} - t^{k,n}(p_m)\right\} \quad \forall p_m, n \tag{8}$$

This can be implemented by selecting the best assignment among the available nodes starting from the first segment of any requested video. We can rank the mapping of all the marginal gains and assign them in decreasing order. The best assignment will be the one allowing to have the highest marginal gain, hence minimizing the outage probability.

As shown in Algorithm 1, the algorithm works by processing over all the nodes (line 1). It is initialized by reading the video request table, the related video for each user as well as the initial segments allocation map $\mathbf{A}^0$ (lines 2–4). These data are used for evaluating the delays of all the possible placements of the segments in the allocation map on all the fog nodes in the network (lines 5–9). This evaluation is used for changing the placement of the chunks for maximizing the marginal gain of each segment (lines 11–14). The algorithm has a computational complexity of $O(N \times P_m \times K)$.

Algorithm 1 Marginal gain maximization.

```
Input: $\mathcal{U}, \mathcal{V}, \mathcal{F}, \mathbb{P}$
Output: $\mathbf{A}$
 1: for all $n \in \mathcal{U}$ do
 2:     Read the Video Request Table $\mathbb{P}$
 3:     Select the video $m$ requested by the user $n$
 4:     Read the segments Allocation Map $\mathbf{A}^0$
 5:     for all $p_m \in \mathbf{A}^0$ do
 6:         for all $k \in \mathcal{F}$ do
 7:             Calculate $\delta(n,k,p_m) = \frac{d^q_{p_m}}{VBR(q)} - t^{k,n}(p_m)$
 8:         end for
 9:     end for
10: end for
11: for $p_m \in \mathbf{A}^0$ do
12:     Find $\bar{k} = \arg\max \delta(n,k,p_m)$
13:     $A(\bar{k}, p_m) = q$
14: end for
```

The second algorithm, named *quality downgrade*, is based on (8) to check the marginal gain for every user having its own segments on any fog node. For any segment not respecting the constraint, the quality related to it will be reduced so as to respect the latency constraint. To this aim, a proper decision is taken based on the evaluation of the latency outage, defined as:

$$\delta(n,k,p_m) = \sum_{p_m=1_m}^{\bar{p}_m} t^{k,n}(p_m) - \sum_{p_m=1_m}^{\bar{p}_m} \frac{d^{\bar{q}p_m}}{VBR(\bar{q})} \tag{9}$$

The goal of the algorithm is to offer an uninterrupted stream, even if it means providing segments with a lower quality than the user requested. This is more suitable in time-sensitive applications where the stream interruption would be avoided.

As shown in Algorithm 2, it is possible to notice that each time the marginal gain is calculated, in the case that the latency outage in (9) is not respected, the quality is downgraded up to the lower level, i.e., $q = 1$, and the map allocation table updated accordingly. The computational complexity of the algorithm is $O(N \times P_m \times K \times X)$, where X stands for the number of iterations in the while loop.

Algorithm 2 Quality downgrade.

```
Input: $\mathcal{U}, \mathcal{V}, \mathcal{F}, \mathbb{P}$
Output: $\mathbf{A}, Q$
 1: for $n = 1$ to $N$ do
 2:     Read the Video Request Table $\mathbb{P}$
 3:     Select the video $m$ requested by the user $n$
 4:     Read the segments Allocation Map $\mathbf{A}^0$
 5:     for all $p_m \in \mathbf{A}^0$ do
 6:         for all $k \in \mathcal{F}$ do
 7:             Calculate $\delta(n,k,p_m) = \frac{d^q_{p_m}}{VBR(q)} - t^{k,n}(p_m)$
 8:             while $\delta(n,k,p_m) < 0$ and $A(k,p_m) \neq 1$ do
 9:                 $A(k,p_m) = A(k,p_m) - 1$
10:                 Update $\delta(n,k,p_m) = \frac{d^q_{p_m}}{VBR(q)} - t^{k,n}(p_m)$
11:             end while
12:         end for
13:     end for
14: end for
15: for $p_m \in \mathbf{A}^0$ do
16:     Find $\bar{k} = \arg\max \delta(n,k,p_m)$
17:     $A(\bar{k},p_m) = q$
18: end for
```

In addition, a playout rate-based buffering technique, while taking advantage of the position of the segments and the length of the video, works in parallel with any of the previous algorithms by adapting the buffering on the playout rate so that the segments downloaded first will belong to the first portion of the video. In this way, the video can be loaded faster and the rules can be relaxed for the remaining segments of the video.

This is favourable in slow networks in which the method allows to stream the first part of the videos as fast as it can to permit the users to start watching and then it downloads the rest while the user is occupied with watching, but it can be detrimental if the users want to skip the first part, as that will lead to a waste of bandwidth.

The two previously introduced algorithms have been compared with two other baseline algorithms in order to better understand their advantages and possible issues. In the first one, named *fixed allocation*, the segments of the videos on the fog nodes are considered to be fixed throughout the whole evaluation. Taking into account (1), the segments are distributed over the allocation matrix to K nodes handling P segments distributed for M videos. Each node is able to handle a maximum number of segments, leaving some segments not available and having to request to the cloud for downloading the required segment. This is the simplest algorithm, and it can be executed without much additional cost, but it does leave room for improvement. It is a general-purpose algorithm that can be most beneficial for fast networks. The second benchmark is instead named *optimal buffering*. In this case, instead, the goal was to reallocate the segments originally placed so as to respect both (5) and the requested quality. However, in order to avoid having unfeasible solutions while minimizing **P1**, we consider that the quality should always follow the one requested.

6. Numerical Results

Numerical results were obtained in MATLAB. We created a network that consists of videos of different lengths segmented over fog nodes and accessed by users. The following comparisons were performed to evaluate the performance of the proposed algorithm when different parameters are changed. To this aim, we considered changing the number of users, while keeping the number of fog nodes and videos fixed, changing the number of fog nodes while keeping the number of users and videos fixed, and changing the number of videos while keeping the number of users and fog nodes fixed.

We hypothesized having a working area with a size of 100 × 100 m. The data rate among nodes was supposed to be equal to 150 Mb/s at a distance of 10 m, though it decreased as the distance increased according to (4). The data rate for downloading from the cloud was set at a fixed rate equal to 15 Mb/s.

There were 5 quality levels corresponding to different video rates, equal to 1 MB/s, 2 MB/s, 5 MB/s, 10 MB/s and 20 MB/s, respectively. The transcoding time was considered a function of the difference between the target and the original quality level. In particular, we set $\delta_{tc}(d^{q}_{pm}, d^{\bar{q}_n}_{pm}) = 0.1(q - \bar{q})$, corresponding to say that we need 0.1 s for each quality level to be changed. The duration of each segment was a random value between 1 s and 7 s [29]. The system was evaluated through the utility function:

$$U(x) = log(x + 1) \tag{10}$$

where x maps the quality of the requesting user. The users were then clustered based on the quality of the segments they request. The higher the number returned by the function, the better the performance of the system [28]. Since the grouping is performed on the quality, $U(x)$, it corresponds to measuring how well the system is performing and is affected by the change in quality. The utility function was used since it is a simple way to assess the behaviour of the system.

Three scenarios for evaluating the performance were set up, as detailed in Table 1. When varying the users, an evaluation was performed by having a network that consisted of 10 fog nodes and 10 videos; the number of users increased from 2 users to 100 users. When varying the nodes, the evaluation was performed by having a network that consisted of 20 users and 10 videos; the number of fog nodes increased from 3 fog nodes to 20 fog nodes. When varying the videos, the evaluation was performed by having a network that consisted of 20 users and 10 fog nodes; the number of videos increased from 5 videos to 25 videos. In all these cases, the videos can be short (i.e., 10 segments), medium (i.e., 100 segments) or long (i.e., 1000 segments). In order to have a fair comparison among all the three video length cases, we fixed the capacity of each for g node equal to 20% of all the segments of all the videos present in the system.

Table 1. Simulation Parameters.

	Users	Fog Nodes	Videos	Segments
Varying Users	{2, 5, 10, 20, 50, 100}	10	10	10 For Short Videos
Varying Nodes	20	{3, 5, 10, 15, 20}	10	100 For Medium Videos
Varying Videos	20	10	{5, 10, 15, 20, 25}	1000 For Long Videos

6.1. Variable Users

The first set of results refer to the scenario where the users are changed. In addition, all three different length types of videos were considered. Three type of results were considered for testing the performance of the proposed solutions. In the first one, represented in Figure 2, the average downloading time for the whole videos is represented. It is possible to notice that, as expected, the longer the videos are, the higher the downloading time is. The quality downgrade has, in general, the lowest downloading time, except for the case of a reduced number of users. Despite this seeming a good result, we have to stress that in the downgrade condition, we would like to minimize the downloading time at the cost of reducing the quality of the videos requested. It is also interesting to notice that the optimal buffering has a video downloading time that is higher than the marginal gain maximization. Additionally, for this case, as becomes more clear by looking at the following figures, we have to consider that in optimal buffering, we would prefer to respect the buffering constraints and not merely minimize the marginal gain.

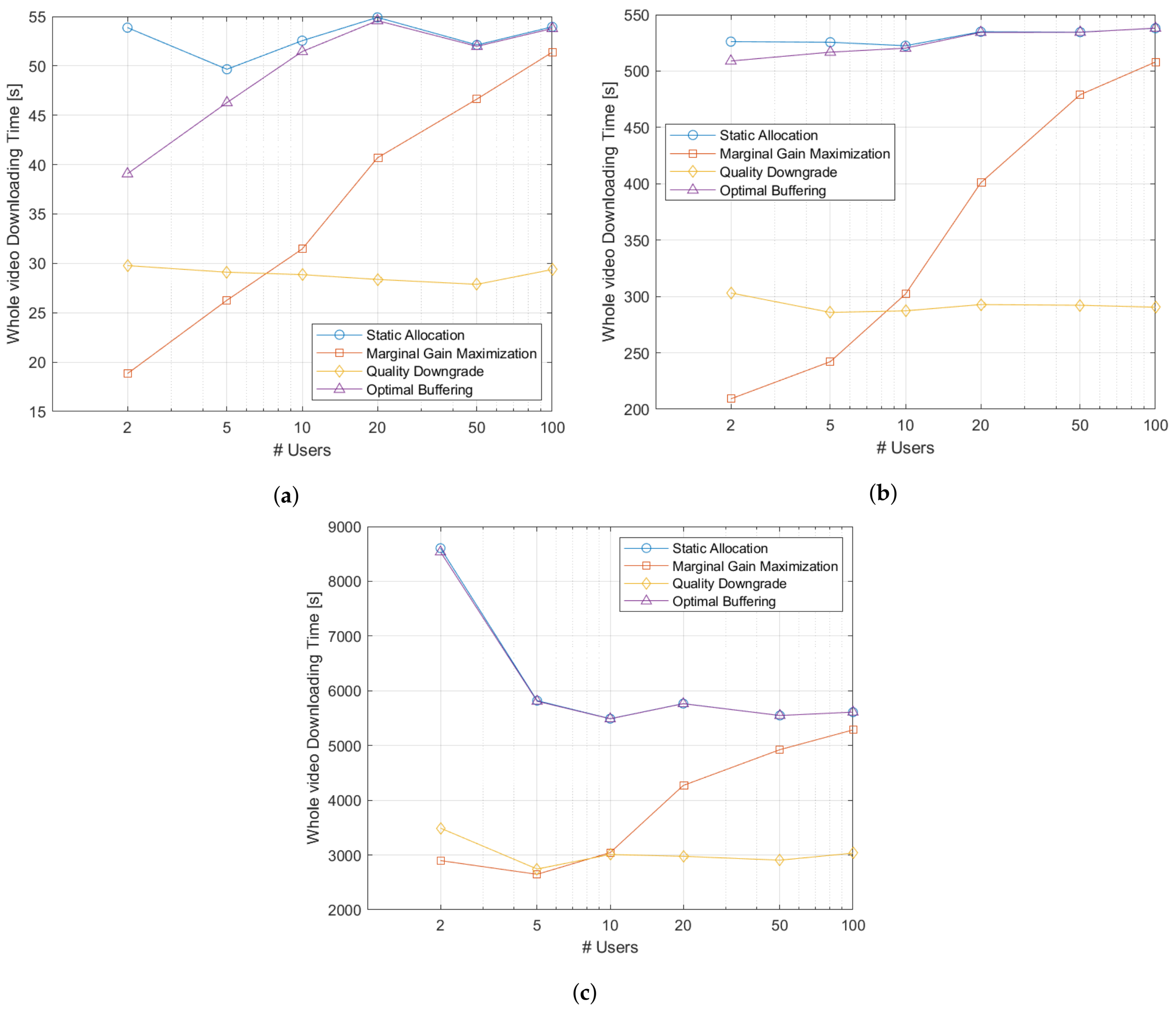
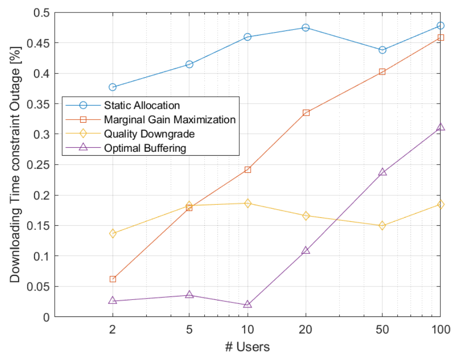


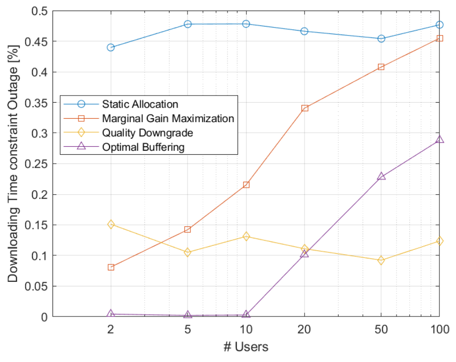
Figure 2. Average video downloading time with variable number of users. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.

Indeed, by analyzing the probability of not respecting the downloading time constraint, i.e., incurring in a potential under-buffering, as depicted in Figure 3, it is possible to notice that the Optimal buffering allows the outage probability to be reduced as much as possible. As we can notice, once the number of users increases, there is some possibility that the quality downgrade performs better. However, once again, we have to stress that in this case, the quality is reduced, while in all the other cases, the quality is always that requested by the users. Regarding the marginal gain, we can notice that it is able to perform better than the static allocation, while the optimal buffering has better performance.

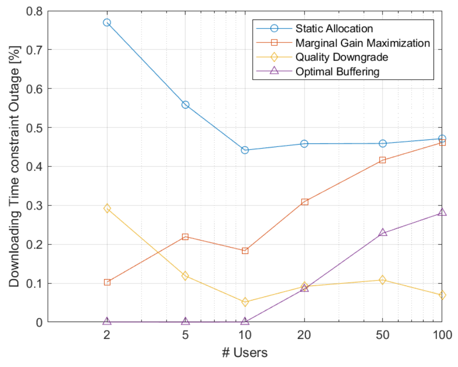
As a third result, we show the utility function (Figure 4). In this case, we cannot be surprised. The downgrade reduces the quality requested by the users; hence, the related utility function is lower, while in all the other cases, the quality is not changed and the same result occurs.



(**a**)

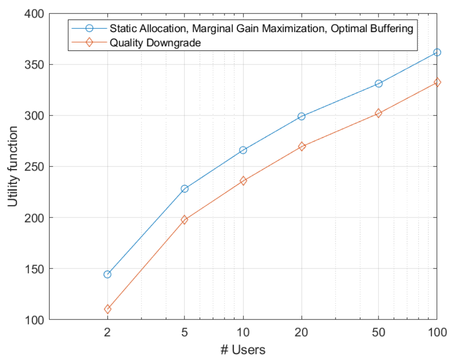


(**b**)

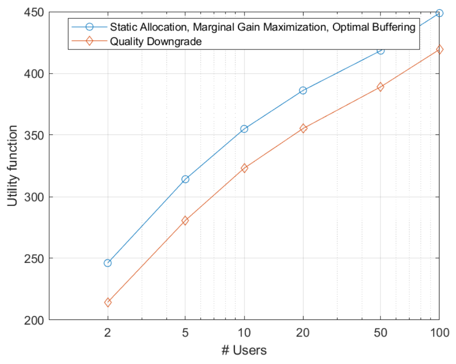


(**c**)

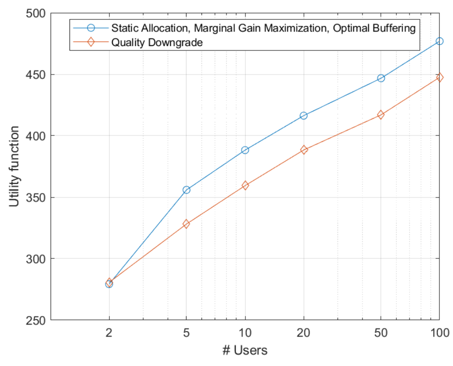
Figure 3. Outage probability with respect to downloading time constraints with a variable number of users. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.



(**a**)



(**b**)



(**c**)

Figure 4. Utility function with variable number of users. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.

Finally we show the probability of quality downgrade events. They are calculated as the quality downgrade levels for each transmitted segment by the whole number of segments. Here everything become clear by noticing that the quality downgrade time is reduced at the cost of having not respected the requested quality. In particular, when the number of users is high, several segments do not respect the requested quality even if the probability of delivering them in time is higher, as previously shown. The values are reported in Table 2.

Table 2. Probability of quality downgrade events with variable number of users.

# Users		**Probability**	
	Short	**Medium**	**Long**
2	0.012	0.014	0.023
5	0.032	0.035	0.044
10	0.068	0.07	0.06
20	0.144	0.138	0.14
50	0.344	0.346	0.36
100	0.719	0.707	0.71

6.2. Variable Nodes

Similar results can be seen in Figures 5–7. As expected, when the number of nodes increases, the performance becomes better. Similar to the previous case, the optimal buffering algorithm allows us to minimize the outage probability, and despite having an overall video download time higher the the rest, it is sufficient for avoiding under-buffering. The marginal gain maximization allows again to trade-off by reducing the download time while having less outage than the static allocation.

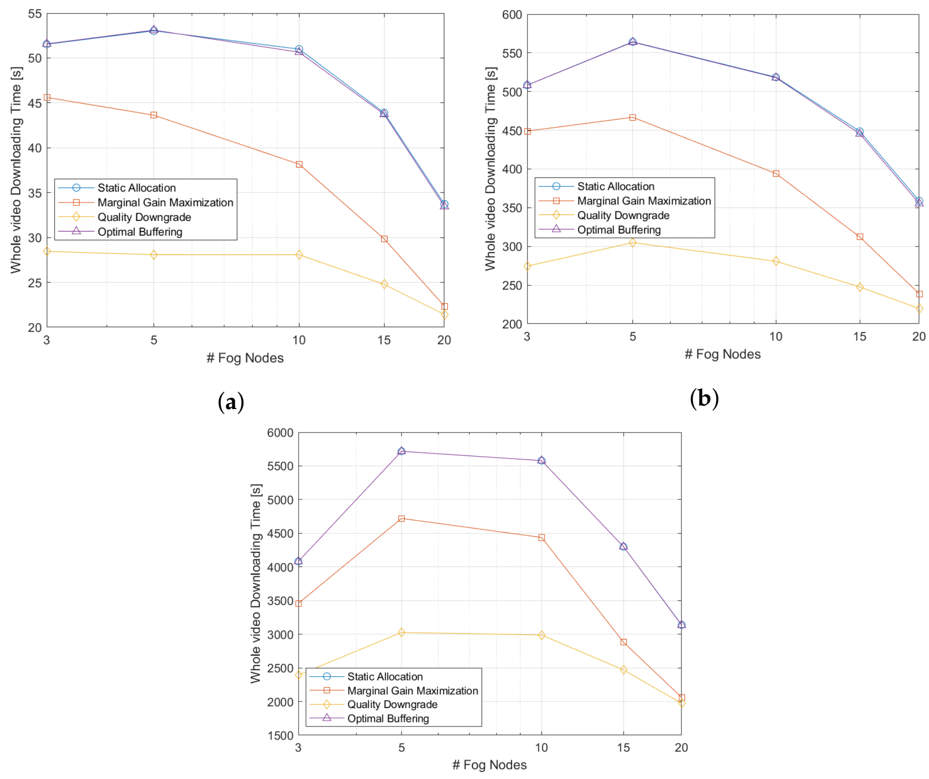
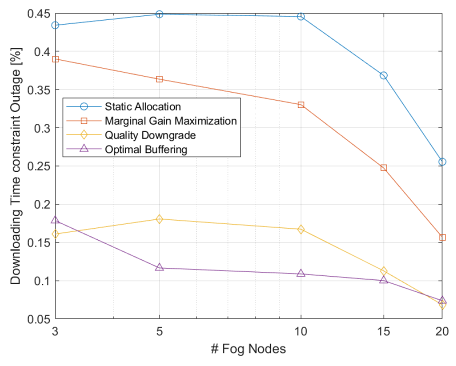
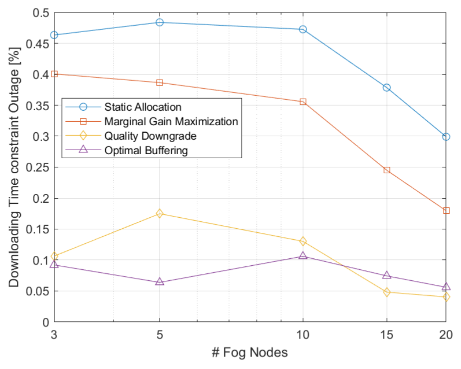


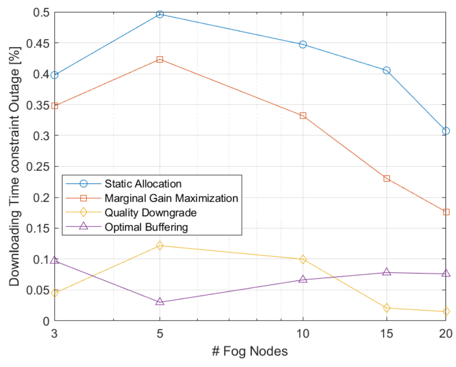
Figure 5. Average video downloading time with variable number of nodes. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.



(**a**)

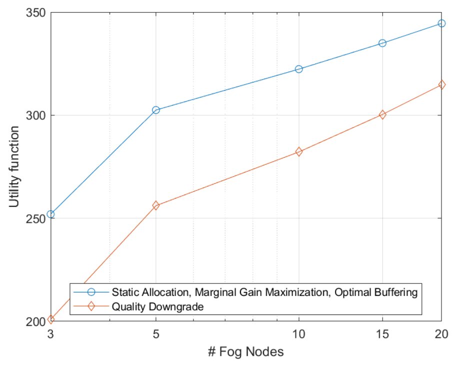


(**b**)

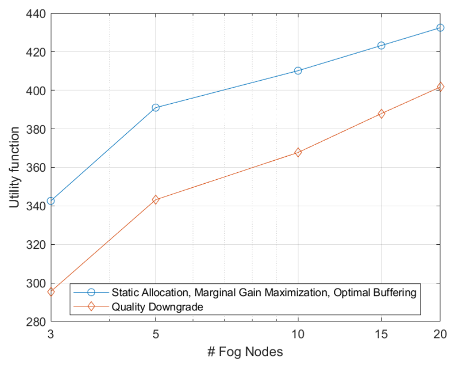


(**c**)

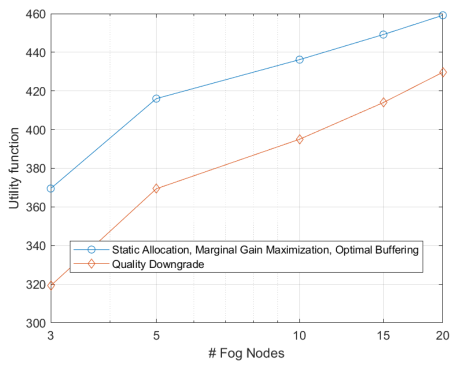
Figure 6. Outage probability with respect to downloading time constraint with variable number of nodes. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.



(**a**)



(**b**)



(**c**)

Figure 7. Utility function with variable number of nodes. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.

A similar trend can be also seen by analyzing Table 3, where we report the probability of downgraded quality levels. As can be seen, the downgrade events are still significant and are needed for keeping the downloading time low. Additionally, when the number of nodes increases, the downgrade events decrease, gaining from the presence of multiple nodes and hence reducing the probability of requesting segments to the cloud.

Table 3. Probability of quality downgrade events with variable number of nodes.

# Nodes	**Probability**		
	Short	**Medium**	**Long**
3	0.6	0.633	0.515
5	0.72	0.753	0.761
10	0.665	0.708	0.705
15	0.569	0.565	0.58
20	0.411	0.458	0.445

6.3. Variable Videos

Finally, we considered the performance in the case that the number of videos stored in the system changes (Figures 8–10). In this case, we can notice that the downloading performance is quite flat for all the algorithms, except the marginal gain maximization. This is due to the fact that when the number of videos increases, the marginal gain can better exploit the reduced ration in terms of users per video, hence decreasing the time needed for downloading them. Similar results to the previous cases can be seen when analyzing both the outage probability and the utility Function.

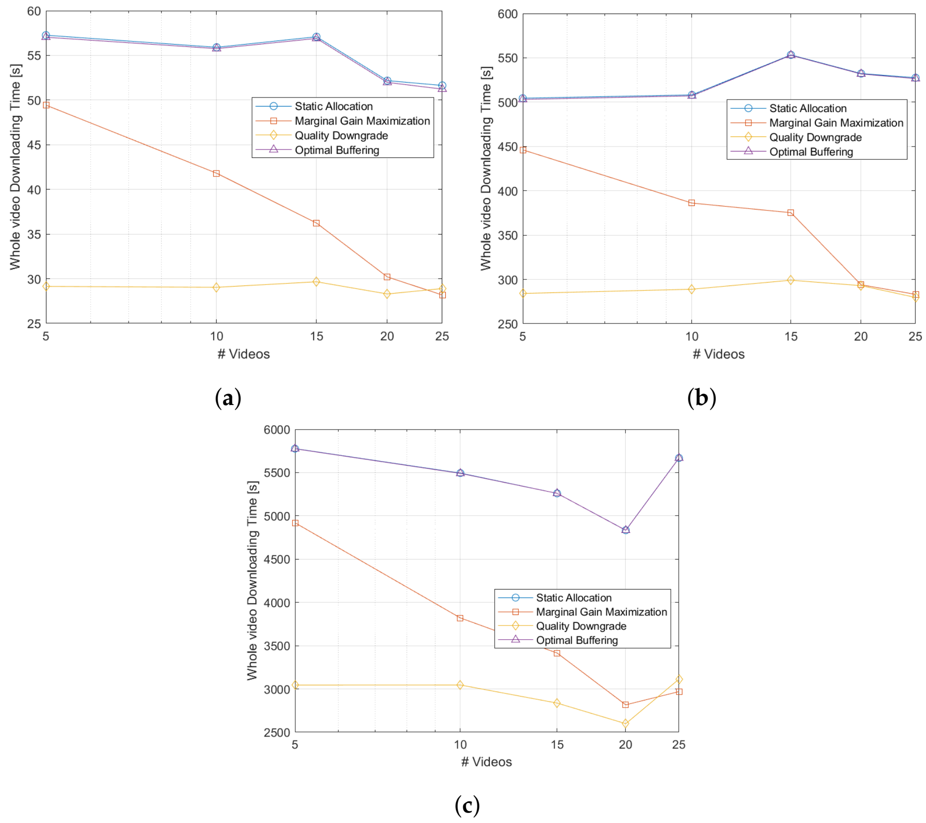


Figure 8. Average video downloading time with variable number of videos. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.

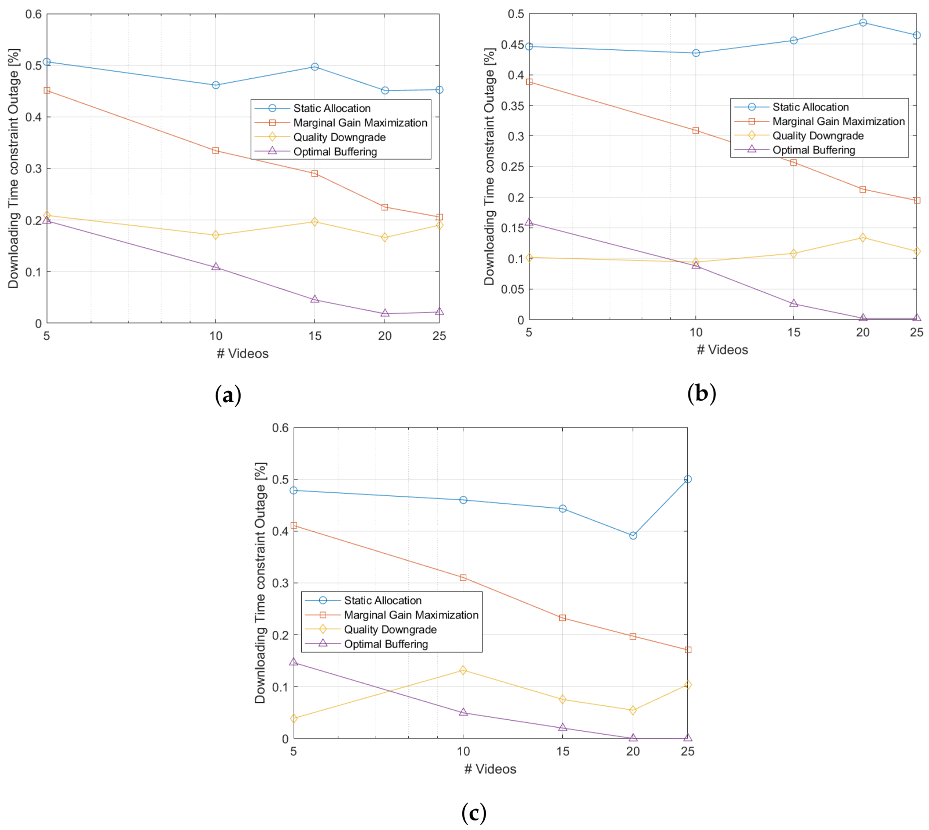


Figure 9. Outage probability with respect to downloading time constraint with variable number of videos. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.

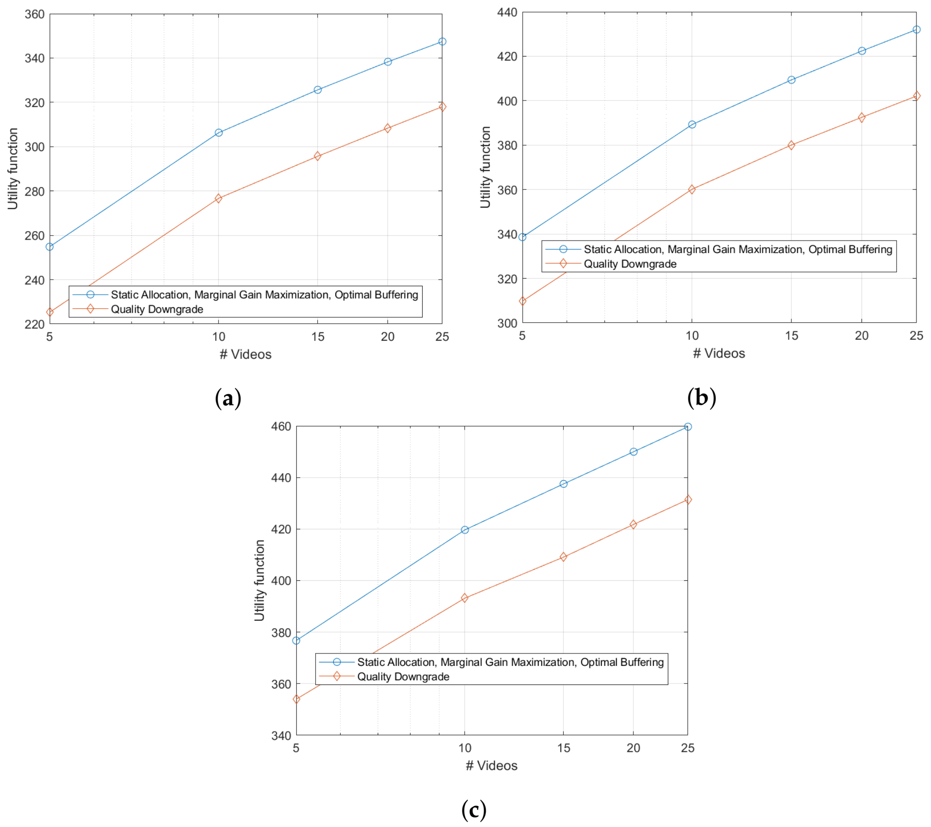


Figure 10. Utility function with variable number of videos. (**a**) Short videos. (**b**) Medium videos. (**c**) Long videos.

A similar trend can also be seen by analyzing Table 4, where we report the probability of downgrades in quality level. As can be seen, the downgrade events are still significant and are needed for keeping the downloading time low. Additionallywhen the number of videos increases the downgrade events decreases, gaining from the reduced number of users request per video, allowing better allocation for each of them.

Table 4. Probability of quality downgrade events with variable number of videos.

# Videos	**Probability**		
	Short	**Medium**	**Long**
5	0.769	0.663	0.69
10	0.754	0.632	0.699
15	0.77	0.685	0.668
20	0.69	0.718	0.628
25	0.672	0.696	0.733

7. Discussion

Fog computing can be seen as a supplement to cloud computing. By extending the cloud and capitalizing on it, we were able to reduce the download time by relying on the segments existing on the fog nodes to stream videos and only using the cloud when the required segments are missing. Computation is done on multiple fog nodes, reducing the processing effort and allowing the placement of the segments in closer locations, therefore decreasing the latency. The DASH protocol used in the fog node breaks up the videos into small segments available in different qualities and allocates them to the fog nodes, allowing a better quality and boosting the performance of the system. The results show reduced latency, improved quality and enhanced system performance by using fog-computing-assisted DASH following different algorithms. The static allocation algorithm distributes the segments of the videos over the fog nodes in an arbitrary way and was used as a basis for the comparison with other algorithms. The marginal gain maximization algorithm rearranges the assignment of the segments of the videos through the fog nodes using the segments' marginal gain and ranking them. This has improved the QoE by ensuring less buffering and less download time. The downgrade algorithm moves the segments to a lower quality if they cannot meet the requirements to avoid buffering by grouping segments of similar qualities and inspecting them, which has improved the QoE by eliminating buffering and reducing the download time. The optimal buffering algorithm allows us to allocate the segments so that the under-buffering is minimized while always respecting the required quality by the users.

Author Contributions: Conceptualization, D.T.; methodology, D.T.; software, M.N. and D.T; validation, M.N., A.B. and D.T.; investigation, M.N., A.B. and D.T.; data curation, M.N. and D.T.; writing—original draft preparation, M.N.; writing—review and editing, A.B. and D.T.; supervision, D.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mehrabi, A.; Siekkinen, M.; Ylä-Jaaski, A. QoE-Traffic Optimization Through Collaborative Edge Caching in Adaptive Mobile Video Streaming. *IEEE Access* **2018**, *6*, 52261–52276. [CrossRef]
2. Sodagar, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia* **2011**, *18*, 62–67. [CrossRef]
3. Bross, B.; Chen, J.; Ohm, J.R.; Sullivan, G.J.; Wang, Y.K. Developments in International Video Coding Standardization after AVC, with an Overview of Versatile Video Coding (VVC). *Proc. IEEE* **2021**, *109*, 1463–1493. [CrossRef]
4. Boyce, J.M.; Ye, Y.; Chen, J.; Ramasubramonian, A.K. Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 20–34. [CrossRef]
5. Lawton, G. Cloud Streaming Brings Video to Mobile Devices. *Computer* **2012**, *45*, 14–16. [CrossRef]
6. Mukherjee, M.; Shu, L.; Wang, D. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1826–1857. [CrossRef]
7. Bozorgchenani, A.; Disabato, S.; Tarchi, D.; Roveri, M. An energy harvesting solution for computation offloading in Fog Computing networks. *Comput. Commun.* **2020**, *160*, 577–587. [CrossRef]
8. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2322–2358. [CrossRef]
9. Bozorgchenani, A.; Tarchi, D.; Corazza, G.E. Centralized and Distributed Architectures for Energy and Delay Efficient Fog Network-Based Edge Computing Services. *IEEE Trans. Green Commun. Netw.* **2019**, *3*, 250–263. [CrossRef]
10. Veillon, V.; Denninnart, C.; Salehi, M.A. F-FDN: Federation of Fog Computing Systems for Low Latency Video Streaming. In Proceedings of the 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC), Larnaca, Cyprus, 14–17 May 2019; pp. 1–9. [CrossRef]
11. Bilal, K.; Erbad, A. Edge computing for interactive media and video streaming. In Proceedings of the 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain, 8–11 May 2017; pp. 68–73. [CrossRef]
12. He, Q.; Zhang, C.; Ma, X.; Liu, J. Fog-Based Transcoding for Crowdsourced Video Livecast. *IEEE Commun. Mag.* **2017**, *55*, 28–33. [CrossRef]
13. Barz, H.; Bassett, G. *Multimedia Networks: Protocols, Design, and Applications*; John Wiley & Sons, Ltd.: Chichester, UK, 2016. [CrossRef]
14. Milovanovic, D.A.; Bojkovic, Z.S. An Evolution of 5G Multimedia Communication: New Ecosystem. In *5G Multimedia Communication: Technology, Multiservices, and Deployment*; Bojkovic, Z.S., Milovanovic, D.A., Fowdur, T.P., Eds.; CRC Press: Boca Raton, FL, USA, 2020.
15. Mazza, D.; Tarchi, D.; Corazza, G.E. A Unified Urban Mobile Cloud Computing Offloading Mechanism for Smart Cities. *IEEE Commun. Mag.* **2017**, *55*, 30–37. [CrossRef]
16. Yoon, C.S.; Jung, H.S.; Park, J.W.; Lee, H.G.; Yun, C.H.; Lee, Y.W. A Cloud-Based UTOPIA Smart Video Surveillance System for Smart Cities. *Appl. Sci.* **2020**, *10*, 6572. [CrossRef]
17. Felici-Castell, S.; García-Pineda, M.; Segura-Garcia, J.; Fayos-Jordan, R.; Lopez-Ballester, J. Adaptive live video streaming on low-cost wireless multihop networks for road traffic surveillance in smart cities. *Future Gener. Comput. Syst.* **2021**, *115*, 741–755. [CrossRef]
18. Ji, W.; Xu, J.; Qiao, H.; Zhou, M.; Liang, B. Visual IoT: Enabling Internet of Things Visualization in Smart Cities. *IEEE Network* **2019**, *33*, 102–110. [CrossRef]
19. Kunst, R.; Avila, L.; Pignaton, E.; Bampi, S.; Rochol, J. Improving network resources allocation in smart cities video surveillance. *Comput. Networks* **2018**, *134*, 228–244. [CrossRef]
20. Jiang, X.; Yu, F.R.; Song, T.; Leung, V.C.M. A Survey on Multi-Access Edge Computing Applied to Video Streaming: Some Research Issues and Challenges. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 871–903. [CrossRef]
21. do Prado, P.F.; Peixoto, M.L.M.; Araújo, M.C.; Gama, E.S.; Gonçalves, D.M.; Silva, M.V.S.; Immich, R.; Madeira, E.R.M.; Bittencourt, L.F. Mobile Edge Computing for Content Distribution and Mobility Support in Smart Cities. In *Mobile Edge Computing*; Mukherjee, A., De, D., Ghosh, S.K., Buyya, R., Eds.; Springer International Publishing: Cham, Switzerland , 2021; pp. 473–500. [CrossRef]
22. Wang, C.; Zink, M. On the Feasibility of DASH Streaming in the Cloud. In Proceedings of the Network and Operating System Support on Digital Audio and Video Workshop, Singapore, 19–20 March 2014; pp. 49–54. [CrossRef]
23. Ma, H.; Seo, B.; Zimmermann, R. Dynamic Scheduling on Video Transcoding for MPEG DASH in the Cloud Environment. In Proceedings of the 5th ACM Multimedia Systems Conference, Association for Computing Machinery, Singapore, 19 March 2014; pp. 283–294. [CrossRef]
24. Gao, G.; Zhang, W.; Wen, Y.; Wang, Z.; Zhu, W. Towards Cost-Efficient Video Transcoding in Media Cloud: Insights Learned from User Viewing Patterns. *IEEE Trans. Multimed.* **2015**, *17*, 1286–1296. [CrossRef]
25. Santos, H.; Alencar, D.; Meneguette, R.; Rosário, D.; Nobre, J.; Both, C.; Cerqueira, E.; Braun, T. A multi-tier fog content orchestrator mechanism with quality of experience support. *Comput. Netw.* **2020**, *177*, 107288. [CrossRef]
26. Maiti, P.; Shukla, J.; Sahoo, B.; Turuk, A.K. QoS-aware fog nodes placement. In Proceedings of the 2018 4th International Conference on Recent Advances in Information Technology (RAIT), Dhanbad, India, 15–17 March 2018. [CrossRef]
27. Lee, G.; Saad, W.; Bennis, M. An Online Optimization Framework for Distributed Fog Network Formation with Minimal Latency. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 2244–2258. [CrossRef]

28. Wang, D.; Peng, Y.; Ma, X.; Ding, W.; Jiang, H.; Chen, F.; Liu, J. Adaptive Wireless Video Streaming Based on Edge Computing: Opportunities and Approaches. *IEEE Trans. Serv. Comput.* **2019**, *12*, 685–697. [CrossRef]
29. Zach, O.; Slanina, M. Content Aware Segment Length Optimization for Adaptive Streaming over HTTP. *Radioengineering* **2018**, *27*, 819–826. [CrossRef]
30. Sharma, P.K.; Singh, R.K. Comparative analysis of propagation path loss models with field measured data. *Int. J. Eng. Sci. Technol.* **2010**, *2*, 2008–2013. Available online: http://www.ijest.info/docs/IJEST10-02-06-85.pdf (accessed on 14 September 2022).
31. *ISO/IEC 23090-3:2020*; Versatile Video Coding, Recommendation H.266. ITU-T: Geneva, Switzerland, 2020.
32. Uhl, T.; Hoppe, C.; Klink, J.H. Modern Codecs by Video Streaming under Use DASH Technique: An Objective Comparison Study. In Proceedings of the 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 17–19 September 2020; pp. 1–5. [CrossRef]