# Video Dissemination in Untethered Edge-Clouds: a Case Study

João Rodrigues, Eduardo R. B. Marques, Joaquim Silva,
Luís M. B. Lopes, and Fernando Silva
{joao.rodrigues,edrdo,joaquim.silva,lblopes,fds}@dcc.fc.up.pt

CRACS/INESC-TEC & Faculty of Science, University of Porto, Portugal

**Abstract.** We describe a case study application for untethered video dissemination using a hybrid edge-cloud architecture featuring Android devices, possibly organised in WiFi-Direct groups, and Raspberry Pi-based cloudlets, structured in a mesh and also working as access points. The application was tested in the real-world scenario of a Portuguese volleyball league game. During the game, users of the application recorded videos and injected them in the edge-cloud. The cloudlet servers continuously synchronised their cached video contents over the mesh network, allowing users on different locations to share their videos, without resorting to any other network infrastructure. An analysis of the logs gathered during the experiment shows that such portable setups can easily disseminate videos to tens of users through the edge-cloud with low latencies. We observe that the edge cloud may be naturally resilient to faulty cloudlets or devices, taking advantage of video caching within devices and WiFi-Direct groups, and of device churn to opportunistically disseminate videos.

## 1 Introduction

Traditional mobile cloud computing focuses on moving processing and storage of data generated by mobile devices to centralised cloud datacenters. This offloading of computation and data benefits the users by decreasing battery consumption in the devices and allows them to access highly reliable infrastructure with seemingly unlimited computational and storage resources. However, due to the distance (both physical and logical) that separates a device at the edge of the network from the cloud, a major technical challenge prevails: how can mobile cloud computing provide applications with low-latency and/or high-bandwidth requirements? What if the infrastructure is unavailable or bandwidth limitations are part of the scenario, e.g., in the aftermath of natural disasters and in dense environments such as sports or music events?

To address these issues, new paradigms such as mobile edge-clouds [3] and cloudlets [18] strategically combine traditional cloud infrastructure with the resources provided by devices and small servers near the edge, enabling proximity-aware applications. In a mobile edge-cloud, for example, nearby devices work

together to form a pool of computing resources with sustained operation under poor connectivity and access to crowd-sourced information which otherwise might be unavailable. Computational tasks are performed locally, i.e., there is no offloading of computation or data to a traditional cloud infrastructure. Cloudlets, on the other hand, bring processing and storage resources closer to the edge to support local offloading of tasks from devices or to serve as caches.

Content Distribution Networks (CDN) can significantly benefit from the aforementioned evolution in edge-cloud technology, e.g., in real world scenarios like sports or concert venues, or social gatherings like weddings, parties and graduation ceremonies. For example, there are apps that provide users within (and outside) sports venues with almost real-time statistics and multimedia contents like the number of kilometres a player has run or video replays for goals or interesting events [15, 26]. Video replays are downloaded from central servers to the mobile devices through the venue's WiFi or cellular infrastructure access points. If, however, the venue is crowded, the large number of requests can stress the infrastructure [4, 7]. In this context, edge-cloud based CDN can be used as a complement to the infrastructure by performing local video dissemination and caching, removing a significant load from the access points. In previous work [19] we showed that this is indeed possible, in a scenario in which a single (venue) server provides the video replays and the user's mobile devices organize themselves into mobile edge-clouds that cache and share those replays. We envision that users can be engaged through diverse incentives, for instance sweepstakes involving goods like team merchandise or game tickets.

In this paper, we explore a more extreme scenario. First, users can consume replays but they can also produce and inject them in the edge-clouds. Second, such clouds of devices can work totally offline, outside the venue's infrastructure, by sharing video replays among themselves and with another tier composed of modest cloudlet servers. The latter synchronize contents periodically allowing for injected videos to be disseminated to different areas of the venue more efficiently. Finally, we use churn, the natural movement of devices in the venue, to disseminate contents opportunistically. We implemented a CDN for the scenario, coupled with an Android app for video acquisition, dissemination and viewing. The software infrastructure was tested in a real world scenario during an official Portuguese volleyball league game. During the experiment, users of the application recorded small video replays of the game through their smartphones and injected them in their edge-cloud and in their local cloudlet servers. The latter synchronised with the other cloudlets in the mesh on a regular basis allowing users on opposite sides of the venue to access videos with different perspectives in almost real-time, and without resorting to any on-site or 3G/4G network.

The rest of the paper is organised as follows. Section 2 presents the scenario and the network architecture to support the "user generated replays" application. Section 3 delves into the implementation of the network tiers. Sections 4 describes the experiment and the results we obtained. Section 5 discusses related work. Section 6 presents concluding remarks and discusses future work.

## 2 Scenario and architecture

Our interest on video dissemination in the context of edge-clouds stems from work in the Hyrax project[1]. The rationale for the latter is to explore a range of, potentially game-changing, crowd-sourcing middleware and applications that harness the collective resources of mobile devices and cloudlets at the edge of the network. Several case study applications were considered in the scope of the project, including caching and device-to-device techniques for video dissemination, distributed computer vision, and fully untethered communications infrastructure for emergency situations.

Previous work in a more restricted scenario, where contents were selected from a TV stream by administrators and published exclusively through central servers, showed that edge-clouds of mobile devices can successfully cache and disseminate a significant fraction of the video contents provided by servers, effectively removing up to 60% of the load from conventional access points [19]. We did this without resorting to rooting the devices or any otherwise intrusive operation that might render the client applications unfit for general public use.

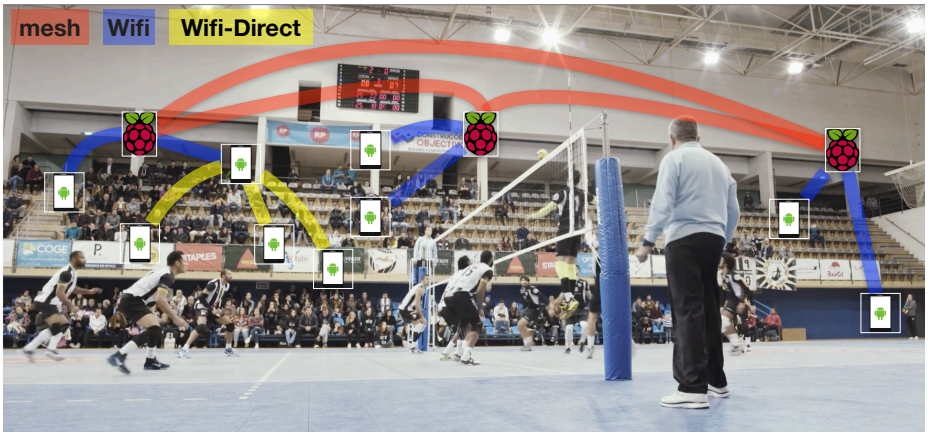



Fig. 1: Snapshot of the game S.C. Espinho vs Vitória S.C., our case study.

In this paper, we explore a more extreme scenario for sports venues (Figure 1) in which users can both consume and produce replays. When they produce a replay they publish it, with some metadata, in the edge-cloud. Also, these clouds of devices can now work totally offline, without the support of the venue's infrastructure. They do this with the help of a second network tier composed of modest cloudlet servers organised in a dynamic peer-to-peer mesh that synchronize their contents on-the-fly. Besides caching the videos published by devices, these servers also work as access points for devices and WiFi-Direct groups (WDGs)

[1] http://www.hyrax.dcc.fc.up.pt

within a given spatial region. Finally, we use churn induced by the natural movement of devices in the venue to disseminate contents opportunistically, allowing devices to publish their videos to neighbouring edge-clouds as they enter them.

Figure 2 shows the 2-tier architecture we used for this scenario. First, in tier 1, we have a mesh of cloudlet servers (in this case Raspberry Pi devices) that actively cache video contents published by tier-2 devices. They feature 2 wireless network interfaces: one to support the cloudlet mesh, the other to provide access points to tier 2 devices. The cloudlet servers actively synchronize their local video caches so that local videos can be accessed by devices under remote cloudlet servers. This is done on a best-effort basis, as no attempt is made to provide any strong form of consistency between the contents of the caches at each cloudlet server. In the absence of network errors (c.f. Section 4) the contents of these caches will progressively converge, i.e., eventual consistency will be attained [22]. Mobile devices in tier 2 can form WDGs and use them to disseminate local contents, or they can connect directly to a cloudlet server. Each device also features a cache for holding the videos it generates and publishes plus the videos it downloads from other devices or cloudlet servers.

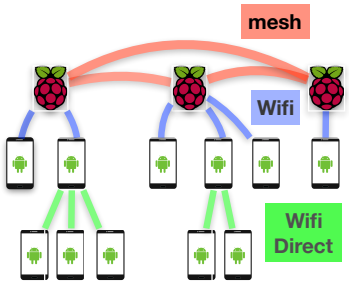
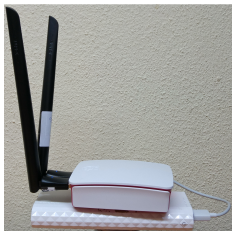



Fig. 2: The 2-tier network architecture.

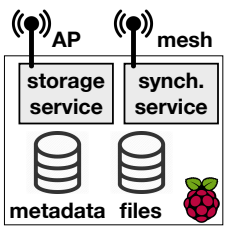
## 3 Implementation

We now describe the main implementation aspects of the infrastructure used in our case-study. We do so in terms of the hardware and software components for cloudlets and mobile devices, and the main algorithms used for video dissemination and edge cloud formation.

**Cloudlets.** Our cloudlets are based on Raspberry 3 Model-B minicomputers, shown in Fig. 3a. These are equipped with a quad-core ARM Cortex 1.2 GHz CPU and 1 GB of RAM, and powered through a 20100 mAh TP-Link power bank. The Raspberry has two USB-attached D-Link DWA-172 WiFi cards with omnidirectional antennas, and runs the Raspbian 9.1 distribution with a Linux 4.9 kernel. One of the WiFi cards is setup in



(a) Hardware.




(b) Software services.

Fig. 3: Cloudlet setup.

mesh mode using the BATMAN protocol[2] over a 5 GHz band and a single channel (36) in all cloudlets with 20 MHz width. The other WiFi card is setup as an AP in the 2.4 GHz band using a distinct and non-overlapping 20 MHz channel per cloudlet (a choice of a 5 GHz band for the AP would limit connectivity by legacy AP clients).

Two software services run in each cloudlet, a storage service and a synchronization service, as illustrated in Fig. 3b. Both services are supported by the local filesystem and a MongoDB database, used for storing video files and video metadata, respectively. The storage service, accessible via HTTP, deals with (upload and download) data transfers from mobile devices related to video files or corresponding metadata (video title, creation time, size, etc) and thumbnails (small images used by clients for video preview). The synchronization service is responsible for cloudlet announcement, discovery, and data transfers in the mesh network. Cloudlet announcement and discovery works dynamically over multicast UDP, allowing cloudlets to dynamically join or leave the mesh if necessary, and data transfers are made through TCP sockets.

The algorithm in Figure 4 illustrates how a (possibly dynamic) set of cloudlets synchronize videos over time, using a best-effort active replication scheme that leads to eventual consistency among cloudlets in the absence of network failures. Each cloudlet periodically announces itself over multicast UDP with a hello message, allowing other cloudlets in the mesh to discover it. The hello message has an associated list of identifiers for available videos. As illustrated by the HANDLE_HELLO() procedure, receiving cloudlets determine which videos are not yet locally stored, and add the latter to a queue of pending videos to transfer. This queue is handled by a continuously running procedure, VIDEO_SYNC(), such that each pending video entry is processed by downloading the corresponding metadata, thumbnail, and actual content, that are stored locally afterwards.

```
procedure HANDLE_HELLO(source, video_id_list)
  stored_videos = STORED_VIDEO_IDS()
  new_videos ← DIFF(video_id_list, stored_videos)
  UPDATE_WORK_QUEUE(source, new_videos)
procedure VIDEO_SYNC()
  while RUNNING() do
    (source, video_id) ← WAIT_FOR_WORK()
    metadata ← GET_METADATA(source, video_id)
    thumbnail ← GET_THUMBNAIL(source, video_id)
    video ← GET_VIDEO(source, video_id)
    STORE(video_id, metadata, thumbnail, video)
```

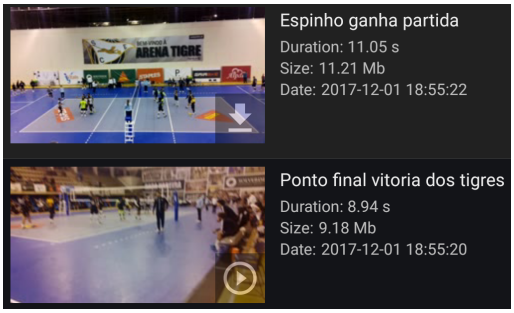
Fig. 4: Cloudlet synchronization algorithm.

**Android app.** The Android app is illustrated in Fig. 5. The app works on non-rooted Android devices running at least Android version 5, thus making use of standard APIs in particular those for WiFi and WiFi-Direct networking. An app user may browse a list of videos (Fig. 5a) identified by their title, creation date, and duration, along with an image thumbnail. Upon selection, a video in this list can be viewed, downloading it first if this has not been done previously, since downloaded videos are stored persistently. The app also allows the reverse operation, i.e., recording a video and uploading it to the network (Fig. 5b). In implementation terms, the app is structured in terms of: a

[2] `https://www.open-mesh.org/projects/open-mesh/wiki`

module that interfaces with the cloudlets' storage service via HTTP; a network manager that administers the dynamic role of the app/device in the network and; an additional module that deals with WDG data transfers via TCP/UDP.

The app may connect to a standard WiFi AP, enabled by one the cloudlets in our scenario, or to a WiFi Direct group enabled by another device working as a "soft" AP, called the group owner (GO), that maintains the WDG and has (is able to maintain) a simultaneous a cloudlet/AP link. In standard operation, data flowing between non-GO members require a network hop through the GO, but Tunnelled Direct Link Setup (TDLS) may also be activated to enable (true) peer-to-peer wireless links [17].




(a) Video browsing.


(b) Video capture.

Fig. 5: Android app.

The dynamics of network formation are discussed further below in this section. Focusing on group interactions for now, a WDG forms a mobile edge cloud where video download requests can be served from within the group, if the videos at stake are available in (i.e., were downloaded previously by) at least one of the members. When not, the GO's cloudlet fulfils the download on-the-fly, either directly, if the request originates at the GO, or with the GO acting as proxy between another group member and the cloudlet. All uploads by non-GO members are similarly routed on-the-fly from the GO to the cloudlet. When proxying file requests, the GO caches file, with the aim of improving resilience to churn in the group and the chance of one-hop transfers in the absence of TDLS links.

Apart from GO caching, and unlike in the cloudlet tier, video replication otherwise occurs passively in the group, i.e., videos are copied between members only when necessary due to an explicit download request by a user. Alternatively, an active replication could imply higher overhead in terms of network bandwidth and inherent battery consumption, but, on the other hand, also potentially lead to faster video dissemination, or reduce user wait-time without much impact in

bandwidth if users tends to watch a high share of a limited pool of videos (e.g., the curated video setting of our previous work [19]).

**Network formation.** Device-to-cloudlet AP connections and WDGs are formed according to the algorithm in Figure 6, running in distributed manner per each device. As shown, four distinct logical connection states are possible, and several event-driven transitions between them: when the device has no active connection (*DISCONNECTED*), it establishes one when there is either an AP or a WDG in range; a device connected to an AP (*AP_CLIENT*) may choose to create or join a WDG; a group owner (*GROUP_OWNER*) may dismantle a group, and resume back to AP client mode and; a group peer (*GROUP_PEER*) may choose to disconnect from the group.

**Parameterisation in the game scenario:**

| $op$ | Creation ($c$) | Disposal ($d$) | Joining ($j$) | Leaving ($l$) |
|---|---|---|---|---|
| $T_{op}$ | 2 | 1 | 5 | 0 |
| $P_{op}$ | 0.5 | 0.5 | 0.7 | 0 |

```
procedure ManageNetwork(T_c, T_d, P_c, P_j)
  // Device thresholds
  input – T_c, T_d, T_j, T_l >= 0:
  // Trans. probabilities
  input – P_c, P_d, P_j, P_l ∈ [0 − 1]:
  state ← DISCONNECTED
  while Running() do
    when ConnectionError() do
      state ← DISCONNECTED
    switch state do
      case DISCONNECTED
        when ∃ap ∈ WiFiNetworks() do
          ConnectTo(ap); state ← AP_CLIENT
        when ∃go ∈ WiFiDirectGroups() do
          ConnectTo(go); state ← GROUP_PEER
      case AP_CLIENT
        when ShouldCreateGroup(T_c, P_c) do
          CreateGroup(); state ← GROUP_OWNER
        when ∃go ∈ WiFiDirectGroups() :
          ShouldJoinGroup(go, T_j, P_j) do
          ConnectTo(go); state ← GROUP_PEER
      case GROUP_OWNER
        when ShouldDisposeGroup(T_d, P_d) do
          DestroyWDG(); state ← AP_CLIENT
      case GROUP_PEER
        when ShouldLeaveGroup(T_l, P_l) do
          Disconnect(); state ← DISCONNECTED
```

Fig. 6: Network formation algorithm.

In the algorithm, state transitions are parameterised by a device threshold $T_{op}$ and a probability $P_{op}$ probability pairs, where *op* may refer to group creation, disposal, joining, or leaving. The threshold parameters impose limits as follows: min. devices visible in the AP network to create a group ($T_c$); min. devices in a group before the GO considers disposal ($T_d$); max. devices that a group may hold, inhibiting other devices to join ($T_j$) and; min. devices in a group before a group peer considers leaving ($T_l$). The probabilities govern the likelihood of transitions, provided all other conditions are enabled for an operation. The transition predicates (e.g. ShouldCreateGroup() for group creation) may also account for runtime conditions in the device and/or the network. Overall, the parameterisation scheme allows for flexible tuning according to the scenario of interest.

The concrete parameters for the game scenario are shown above the listing of Fig. 6, and result from an empirical calibration we did through some preliminary tests at the game venue. They reflect the concerns of not creating WDGs too aggressively ($P_c = 0.5$, $T_c = 3$), trying to maintain them active ($P_d = 0.5$, $T_d = 1$) and stable ($P_l = T_l = 0$) for relatively long, and encouraging devices to join groups and have groups of reasonable size ($P_j = 0.7$, $T_j = 5$). Together

with this parameterisation, the evaluation of group creation, embedded in the implementation of ShouldCreateGroup(), feeds on battery and AP signal strength values broadcasted by devices (when connected to a cloudlet) to implement a simple heuristic: a device becomes GO only if it has the highest value of all $(batteryLevel + signalStrength)/2$ measures known for all devices; the scheme limits the chance that two devices connected to the same cloudlet become GO almost simultaneously. Other transition predicates could also easily be refined to account for runtime conditions, e.g., the battery level of a device while acting as GO in ShouldDisposeGroup() in order to avoid battery depletion (that occurs at a faster rate for GOs, given their role as soft APs), or the signal strength measured for the serving GO to decide leaving a group in ShouldLeaveGroup().

## 4 The experiment

**Setup.** We conducted our real-world experiment during a game of the Portuguese volleyball league between S. C. Espinho and Vitória S. C., that took place on December 1st, 2017, at the Nave Desportiva de Espinho sports venue[3]. We recruited several student volunteers to watch the game and use the video dissemination app, using Google Nexus 9 tablets running Android 6.0 that we provided. Additionally, some audience members and S. C. Espinho staff were also engaged to participate using their own smartphones. In total, 18 users/devices participated in the experiment.

For the experiment, the peer-to-peer mesh formed between the cloudlets was static in size and fully connected. At the venue, we had three cloudlets installed as shown in Figure 1, identified as C1, C2, and C3 in this section. Smartphones and tablets from the volunteers, sitting in the game stands or moving through them, formed tier-2. Before the experiment, we disabled the use of TDLS over WiFi-Direct, as we found that the Android app had stability problems; hence the app used only plain WiFi and WiFi-Direct. Additionally, GPS signal strength/precision was too poor inside the (indoors) sports venue for obtaining good location/mobility logs, hence we instructed volunteers not to activate GPS in the devices.

The experiment began at approximately 16:30, 30 minutes before the game started, and ended just after the game was over at approximately 19:15. During this period, the following types of events were logged for the Android app and cloudlet instance: video data transfers in terms of source, destination, time interval, length of data; network formation events for WiFi-Direct group creation or disposal, and; connection establishment/detachment between devices and cloudlets or WiFi-Direct GOs. Event data was stored locally for the cloudlets and devices, and also pushed from devices to cloudlets in short data transfers. When the experiment was over, we collected and merged all the logs for analysis.

[3] `https://goo.gl/maps/cUainSqtD962`

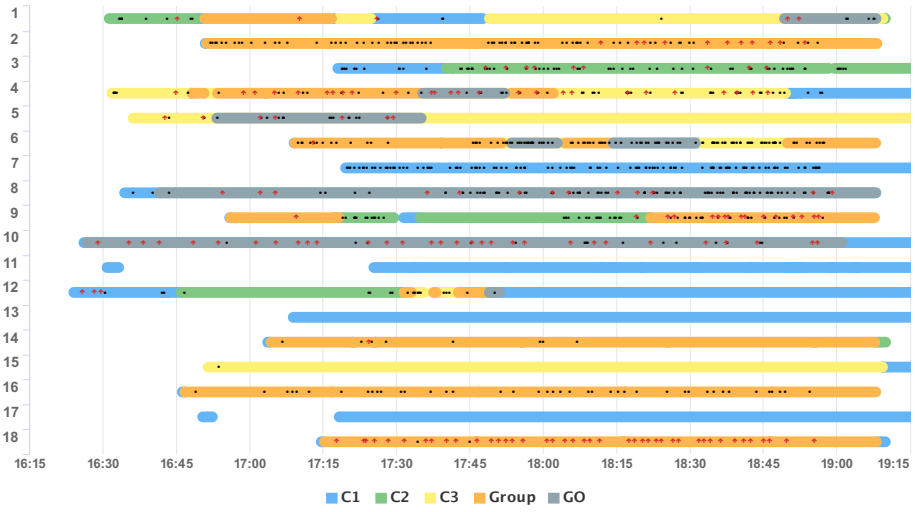


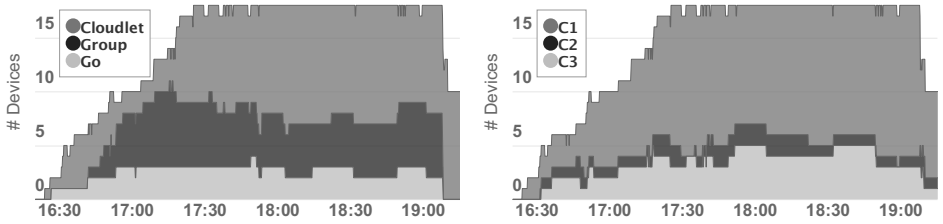

Fig. 7: Timeline of operations per device.

**Overview of user behaviour.** Over the course of the experiment, the volunteers recorded and uploaded 165 videos to the edge cloud, and also performed a total of 660 video downloads. Apart from these successful operations, there were 39 transfer errors, corresponding to approximately 4% of total video transfers.

Figure 7 shows a timeline of operations per device. The colors show, for any given instant, whether a device was connected directly to a cloudlet AP or to a WDG, or if they formed a WDG (acted as GO). For each device timeline, upload and download completions are respectively marked with arrows and dots. As shown, the behaviour observed from device to device can be quite heterogeneous in terms of connectivity and user operations. Some devices connected to a cloudlet AP and remained in that state for most if not all of the game (e.g., 11, 13, 15, 17), others roamed between different cloudlet APs (1, 4, 6, 9, 12), two devices formed WDGs for most of the game (8 and 10), and, finally, a portion of devices was connected to a WDG for significant time (2, 14, 16, 18). Regarding video transfers, some devices were quite active in terms of uploads (10 and 18), downloads (7 and 16), or both (4 and 8), while others had little or no activity (11, 13, 15, 17).

**Connectivity.** Figure 8 depicts stacked graphs for the number of users over time in terms of connection type (8a) and serving cloudlet (Fig. 8b).

First, we can observe that the share of devices that formed WDGs (GO) or connected to them (Group) was roughly balanced with the number of devices that merely connected to cloudlet APs. Over time, 7.6 (49%) devices on average participated in WDGs, 2.5 as group owners (16%) and 5.1 (33%) connected to these groups, whilst 7.9 (51%) devices connected to a cloudlet AP only. The results are less balanced for the serving cloudlet, through direct connection or transitively through a GO: over time 9.8 devices were served by C1 on average,

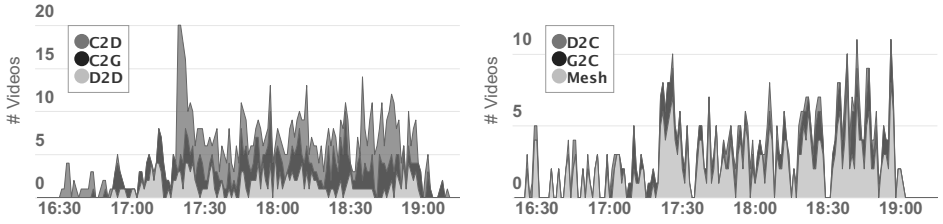



(a) Type of connectivity. (b) Serving cloudlet.

Fig. 8: Connectivity over time.

1.3 by C2, and 3.2 by C3. To explain this, a precise analysis would require accurate user location/mobility data (as mentioned earlier, we were constrained by GPS signal reception) but we attribute this to two facts we observed on-site: (1) users typically installed and turned on the app at a working desk located very near C1, therefore likelier to keep the connection to it longer, and; (2) as suggested by the numbers, there were in fact a higher concentration of users near C1, where most of the game fans concentrated.

**Uploads and downloads.** Figure 9 depicts video transfers in terms of active downloads (9a) and uploads (9b) over 1-minute intervals, as stacked graphs. Downloads are distinguished in terms of the following types of transfers: cloudlet-to-device (C2D), when involving devices that are not part of a WDG; cloudlet-to-group transfers (C2G), for downloads that originated in a WDG but could not be served by it; and device-to-device (D2D), for videos that were requested and served within a WDG. On average, 3.4 C2D (42%), 2.6 C2G (33%), and 2.0 (25%) D2D downloads were active per minute. The C2D share (42%) is slightly less than the share of devices that were not involved in groups over time (51%, as mentioned earlier), whilst the D2D/C2G ratio of 43% corresponds to the proportion of downloads issued and served within the same WDG. Regarding uploads, device-to-cloudlet (D2C) and group-to-cloudlet (G2C) are shown, along with the correlated video transfers in the mesh (using Algorithm 4 presented




(a) Downloads. (b) Uploads.

Fig. 9: Video transfers over time.

earlier). On average, there were 2.5 (1.1 D2C and 1.4 G2C) active uploads and 3.1 mesh transfers active per minute. Note that per each upload, there should be 2 corresponding mesh transfers (we use 3 cloudlets), hence close to 5.0 mesh transfers would be expectable instead. Upon log inspection, we verified that the mesh synchronization got stuck for cloudlet C1 early in the experiment due to a software glitch. Thus, videos originating at devices served by C2 and C3 were not replicated through the mesh onto C1, although cloudlets C2 to C3 functioned properly and were able to pull videos from C1 during the entire experiment.

Looking into this issue further, Figure 10 shows a global plot for the number of distinct videos stored in all 3 cloudlets over time (on top), plus three other plots comparing the number of videos stored per each cloudlet (C$i$) vs. the number of (also distinct) videos stored in the devices served by that cloudlet (A$i$). The global plot clearly indicates that, at around 17:15, C1 started to lag behind C2 and C3. Inspecting the plot for C1, a surprising finding is that the number of videos in devices served by it (A1) did not stop growing, however. In fact, it converged very closely to the set of videos stored in C2 or C3. The behavior of C1 is not observed for C2 and C3 which synchronised over the mesh properly, apart from a temporary glitch in C2 between 17:00 and 17:30, and where the number of videos in served devices was actually much lower than those stored by the cloudlets (owing up to the lower number of users/download requests). The finding for C1 is explained by the combined effects of device churn and video caching: the loss of videos through faulty synchronization at the mesh tier was compensated by devices that eventually got under the scope of C1 and brought most of the missing videos with them. We did not anticipate (the technical glitch or) this possibility, hence C1 was not updated with those videos, but clearly the potential of opportunistic churn-driven synchronization exists and should be seized upon.

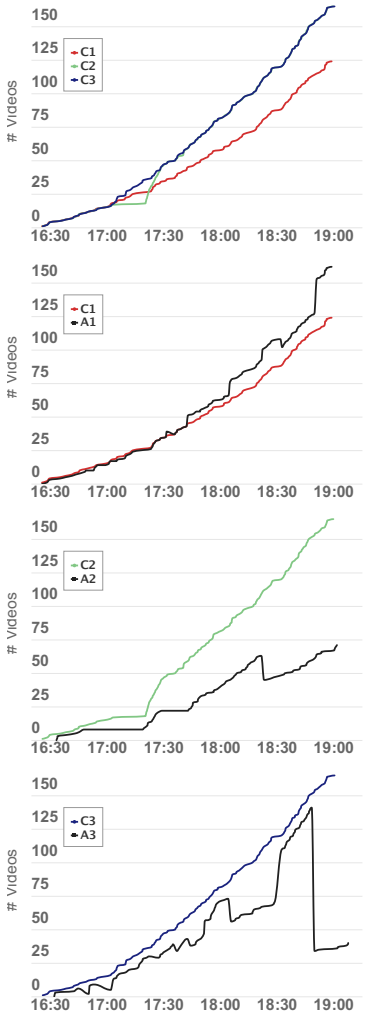
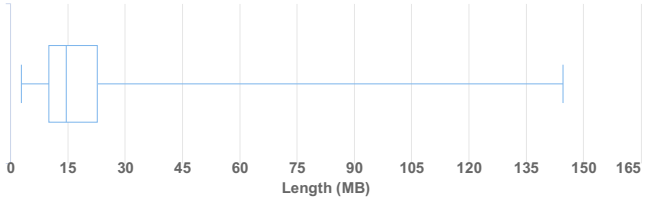
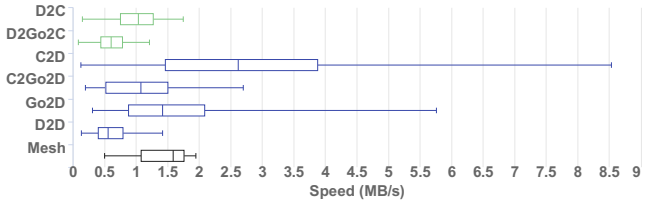


Fig. 10: Video storage analysis.

**Video transfer analysis.** Figure 11 characterises video lengths (11a) and transfer speeds per category (11b), in terms of quartiles, minimum and maximum values. The median size




(a) Size.




(b) Speed.

Fig. 11: Video transfers – size and speed.

for a video was roughly 14.5 MB, and more than 75% were smaller than 25 MB. To put this in perspective, in spite of some larger videos (the largest one had 144 MB), even with low bandwidth, say 500 KB/s, a video transfer typically took less than 1 minute.

The transfer speeds are shown for uploads (green), downloads (blue) and mesh synchronization (red). For uploads, we distinguish between direct device-to-cloudlet (D2C) transfers, and uploads mediated through a GO (D2Go2C), with median values of approximately 1 MB/s and 600 KB/s, respectively. The symmetric values for downloads, C2D and C2GO2D, are higher as expected (downlink speed is higher as usual): 2.6 and 1.1 MB/s. As for downloads made within a WDG, we distinguish transfers from GO to another group member (Go2D) and between non-GO members (D2D), with median values of 1.4 MB/s and 600 KB/s, respectively. D2D transfers are slower, as they require data to be routed through the GO and the inherent up-link bandwidth limitation (the values are similar to the D2Go2C case). TDLS links, disabled for stability reasons, should allow much higher values for D2D transfers (as we have achieved in [17, 19]). Finally, the median value for mesh transfers was 1.6 MB/s.

**WDG analysis.** We now analyse the behavior of the two WDGs that were active for most of the experiment, those enabled by devices 8 and 10 in Figure 7. In comparison, the other 5 WDGs that were formed during the experiment all lasted less than 1 hour. Figure 12 depicts the behavior of the 2 groups in terms of member count (including the GO), the number of videos in the GO, the number distinct videos stored in all members, and the number of total videos stored by the group (including duplicates).

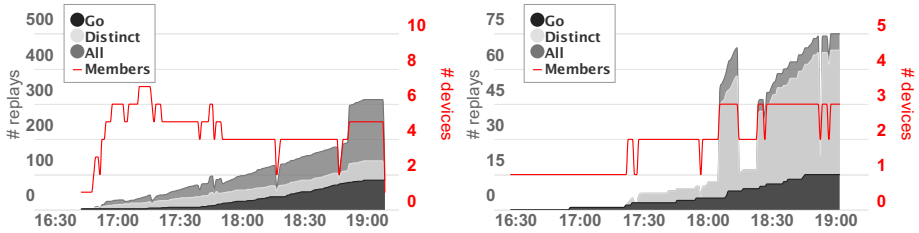



Fig. 12: Analysis of two Wifi-Direct groups.

The first group was clearly more active, with 4.46 members on average and a peak value of more than 300 videos, 122 of which were distinct (74% of all 165 videos in the system). Two other significant traits are that: (1) the GO videos, that are permanently available while the group lasts, grows monotonically over time and stores/caches more than half (52%) of these videos, and; (2) there is a high ratio of replication of videos (1.9), meaning that devices tended to download a high portion of the same videos. This results in resilience to churn: the group's video pool (the set of distinct videos) does not decrease significantly when a member departs, while there are much higher variations in total video count. The second group marks a contrast to the first one, given that it had far few members (1.9), a smaller video pool on average, and low levels of GO caching (27%) and replication (1.1).

## 5 Related work

CDNs are usually materialised by large datacenters spread throughout the globe, emerged with the objective of improving the distribution of content to end-users by providing high data availability, reduced latency and increased network bandwidth. The potential of hybrid CDN-P2P architectures is well acknowledged by large CDN providers, e.g., see [12]. Some architectures combine servers and P2P (Internet) links [5, 24], but do not directly account for proximity to end users. Cloudlets, on the other hand, are lightweight servers deployed at the edge of the network, e.g., as part of hybrid CDN-P2P architectures [25], as relays to perform video processing before uploading data to the central cloud [20], or to deliver geo-based multi-player content and interaction [23]. Mobile devices themselves can be used effectively as CDN caches in the network, even without resorting to D2D links [9].

D2D communications enabled at the level of cellular networks is envisioned as the main enabler for traffic offloading/proximity-aware CDN [1]. While this promise is not fulfilled at scale, WiFi/Bluetooth-based communications are the prevalent means. For instance, Haggle [14] is an opportunistic content-sharing system that employs WiFi and Bluetooth links and store-and-forward techniques to cope with network disruption and churn, and Kwon et al. [10] make use of WiFi-Direct groups for improved video streaming by letting the GO act as relay/cache between the cellular infrastructure and other group members. There

are other non-standard techniques that require "rooted" device extensions to work, e.g., Helgason et al. [6] present a middleware for mobile applications that disseminate contents opportunistically over WiFi ad-hoc one-hop links without further infrastructural support, and Microcast [8] uses D2D WiFi/Bluetooth communication in conjunction with 3G/4G to improve the performance of video streaming, but making use of overhearing techniques for WiFi.

The hybrid architecture of this paper combines cloudlets and D2D communication, and is naturally extensible to the consideration of a centralised cloud layer. In addition to the video dissemination scenario considered here (in sequence to [19], discussed earlier in the paper), another Hyrax project case-study concerns photo sharing for temporary networks formed at social gatherings, making use of the Thyme system [2]. Thyme is a time-aware publish-subscribe CDN service, in which devices are logically organised into geographical hash-tables for content discovery and retrieval. For these and other Hyrax applications and services, a general-purpose middleware [16] is being developed, providing the abstraction of an overlay network formed from heterogeneous (WiFi, WiFi-Direct, Bluetooth) D2D links, and that is adaptive to intermittent communication and device churn.

## 6 Conclusions and future work

We presented an untethered hybrid edge-cloud to support video dissemination at sporting events, validated experimentally in the real world setting of a Portuguese league volleyball game. The edge-cloud was composed of mobile devices, possibly organised in WDGs, that produce and consume videos, and a mesh of three Raspberry Pi cloudlets that cache and disseminate the videos produced by the devices. The experiment showed that the edge cloud was sufficiently robust to provide videos to tens of users with low latencies. Moreover, the multiple caching levels—at devices, WiFi-Direct groups and cloudlets—made it resilient to device or cloudlet failures. In particular, we illustrated that an unexpected long-term fault by one the cloudlets could be compensated through the combined effects of caching and churn for opportunistic content sharing. The role of the WDGs was specially relevant in this overall picture. They significantly offloaded traffic from the mesh infrastructure, by involving 49% of devices on average and serving 43% of the downloads issued by devices in such groups. Moreover, as illustrated for the most active group in the experiment, caching not only provides resilience to churn (when devices leave the group) but in fact seize upon it (when devices enter) to accumulate videos over time in a group.

For future work we focus on several aspects. Some natural extensions can be considered to our hybrid architecture, such as the addition of a centralised cloud layer [19], and the use of 3G/4G networks in combination with WiFi and D2D communication. In the presence of a centralised cloud layer, a curation mechanism can be implemented such that administrative users filter the appropriate videos for dissemination and certify their provenance for security assurances (e.g., using digital signatures for videos). More generally, the paramount

aspects of security and privacy, outside the scope of this paper, may be dealt with by several mechanisms in a mobile edge cloud setting at the level of devices and/or cloudlets, e.g., see [11, 13, 21]. Another line of work concerns the simulation of our scenario, to help us understand the behavior of the network at scale over the parameter space, and calibrate experiments in more principled manner, for instance regarding the strategies for network formation, video caching and replication, patterns for user mobility and video sharing, and choice of communication technologies. These future developments may consider other scenarios in crowded venues (e.g., music halls, museums) or in communication-deprived environments (e.g., disaster settings, remote locations) [16].

## Acknowledgements

This work has been sponsored by projects HYRAX (CMUP-ERI/FIA/0048/2013), funded by FCT, and SMILES (NORTE-01-0145-FEDER-000020), funded by NORTE 2020, under PORTUGAL 2020, and through the ERDF fund. We wish to thank Nuno Vitó and Bernardo Viterbo from S. C. Espinho, José Gouveia and Quirino Gomes from C. M. Espinho, and Francisco Carvalho from Vitória S. C. for their precious support.

## References

1. Andreev, S., Pyattaev, A., Johnsson, K., Galinina, O., Koucheryavy, Y.: Cellular traffic offloading onto network-assisted device-to-device connections. IEEE Communications Magazine 52(4), 20–31 (2014)
2. Cerqueira, F., Silva, J.A., Lourenço, J.M., Paulino, H.: Towards a persistent publish/subscribe system for networks of mobile devices. In: Proc. MECC'17. pp. 2:1–2:6. ACM (2017)
3. Drolia, U., Martins, R., Tan, J., Chheda, A., Sanghavi, M., Gandhi, R., Narasimhan, P.: The Case for Mobile Edge-Clouds. In: Proc. UIC/ATC'13. pp. 209–215. IEEE (2013)
4. Erman, J., Ramakrishnan, K.K.: Understanding the Super-sized Traffic of the Super Bowl. In: Proc. IMC'13. pp. 353–360. ACM (2013)
5. Ghareeb, M., S. Rouibia, B. Parrein, M.R., Thareau, C.: P2PWeb: A Client/Server and P2P Hybrid Architecture for Content Delivery Over Internet. In: Proc. ICCIT'13. pp. 162–166. IEEE (2013)
6. Helgason, O.R., Yavuz, E.A., Kouyoumdjieva, S.T., Pajevic, L., Karlsson, G.: A mobile peer-to-peer system for opportunistic content-centric networking. In: Proc. MobiHeld'10. pp. 21–26. ACM (2010)
7. Kapustka, P., Stoffel, C.: State of the Stadium Technology Survey. Tech. rep., Mobile Sports Report (2014)
8. Keller, L., Le, A., Cici, B., Seferoglu, H., Fragouli, C., Markopoulou, A.: Microcast: Cooperative video streaming on smartphones. In: Proc. MobiSys'12. pp. 57–70. ACM (2012)
9. Koukoumidis, E., Lymberopoulos, D., Strauss, K., Liu, J., Burger, D.: Pocket cloudlets. In: Proc. ASPLOS XVI. pp. 171–184. ACM (2010)

10. Kwon, D., Je, H., Kim, H., Ju, H., An, D.: Scalable video streaming relay for smart mobile devices in wireless networks. PloS one 11(12), e0167403 (2016)
11. Liu, J.K., Au, M.H., Susilo, W., Liang, K., Lu, R., Srinivasan, B.: Secure sharing and searching for real-time video data in mobile cloud. IEEE Network 29(2), 46–50 (2015)
12. Lu, Z., Wang, Y., Yang, Y.R.: An Analysis and Comparison of CDN-P2P-hybrid Content Delivery System and Model. Journal of Communications 7(3), 232–245 (2012)
13. Mollah, M.B., Azad, M.A.K., Vasilakos, A.: Secure data sharing and searching at the edge of cloud-assisted internet of things. IEEE Cloud Computing 4(1), 34–42 (2017)
14. Nordström, E., Rohner, C., Gunningberg, P.: Haggle: Opportunistic mobile content sharing using search. Computer Communications 48, 121–132 (2014)
15. Robb, D.: HuaweiVoice: Agile Stadiums Bring Digital Content To Sports Fans. Forbes Magazine (2015)
16. Rodrigues, J., Marques, E.R.B., Lopes, L.M.B., Silva, F.: Towards a Middleware for Mobile Edge-cloud Applications. In: Proc. MECC'17. ACM (2017)
17. Rodrigues, J., Silva, J., Martins, R., Lopes, L., Drolia, U., Narasimhan, P., Silva, F.: Benchmarking Wireless Protocols for Feasibility in Supporting Crowdsourced Mobile Computing. In: Proc. DAIS'16. pp. 96–108. Springer (2016)
18. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The Case for VM-Based Cloudlets in Mobile Computing. IEEE Pervasive Computing 8(4), 14–23 (2009)
19. Silva, P.M.P., Rodrigues, J., Silva, J., Martins, R., Lopes, L., Silva, F.: Using Edge-Clouds to Reduce Load on Traditional WiFi Infrastructure and Improve Quality of Experience. In: Proc. ICFEC'17. pp. 61–67. IEEE (2017)
20. Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Satyanarayanan, M.: Scalable crowd-sourcing of video from mobile devices. In: Proc. MobiSys'13. pp. 139–152. ACM (2013)
21. Tan, J., Drolia, U., Martins, R., Gandhi, R., Narasimhan, P.: Chips: Content-based heuristics for improving photo privacy for smartphones. In: Proc. WiSec'14. pp. 213–218. ACM (2014)
22. Vogels, W.: Eventually consistent. Communications of the ACM 52(1), 40–44 (2009)
23. Wang, N., Varghese, B., Matthaiou, M., Nikolopoulos, D.S.: ENORM: A framework for edge node resource management. IEEE Transactions on Services Computing (2017)
24. Wang, X., Chen, M., Kwon, T.T., Yang, L., Leung, V.C.M.: AMES-Cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds. IEEE Transactions on Multimedia 15(4), 811–820 (2013)
25. Yin, H., Liu, X., Zhan, T., Sekar, V., Qiu, F., Lin, C., Zhang, H., Li, B.: Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In: Proc. Multimedia. pp. 25–34. ACM (2009)
26. YinzCam. `http://www.yinzcam.com/`, last visited in 21/02/2018