Vision: mClouds – Computing on Clouds of Mobile Devices

Emiliano Miluzzo, Ramón Cáceres, Yih-Farn Chen

AT&T Labs – Research, Florham Park, NJ, USA

ABSTRACT

When we think of mobile cloud computing today, we typically refer to empowering mobile devices – in particular smartphones and tablets – with the capabilities of stationary resources residing in giant data centers. But what happens when these mobile devices become as powerful as our personal computers or more? This paper presents our vision of a future in which mobile devices become a core component of mobile cloud computing architectures. We envision a world where mobile devices will be capable of forming mobile clouds, or *mClouds*, to accomplish tasks locally without relying, when possible, on costly and, sometimes, inefficient backend communication. We discuss a possible mClouds architecture, its benefits and tradeoffs, and the user incentive scheme to support the mCloud design.

Categories and Subject Descriptors: C.2 [Computer-Communication Networks]: Distributed Systems

General Terms: Algorithms, Design

Keywords: Mobile Cloud Computing, Mobile Devices, Distributed Systems

1. INTRODUCTION

The tremendous explosion of mobile devices is far from over. Smartphones and tablets are at the center of a new modern revolution, one that is taking mobile computing to a whole new level. These devices become even smarter with the introduction of novel and powerful services that can take advantage of the computational resources on both the device itself and on the cloud. The game-changing application markets (Apple App Store and Google Android Market), artificial intelligence breakthroughs (AT&T Watson [2], Apple Siri, Google voice search), sensing and communication capabilities are pushing the envelope for a new class of devices towards the realization, one day, of an ambitious grand vision: all-in-one smart, powerful, and versatile devices – the ultimate fusion of phones, tablets, and personal computers. It is not too hard to imagine a world where we will be able to carry a single device to answer phone calls, use it as our primary personal computer at work by hooking it up to external monitors and keyboards, and install our favorite games as much as we do nowadays on tablets and smartphones.

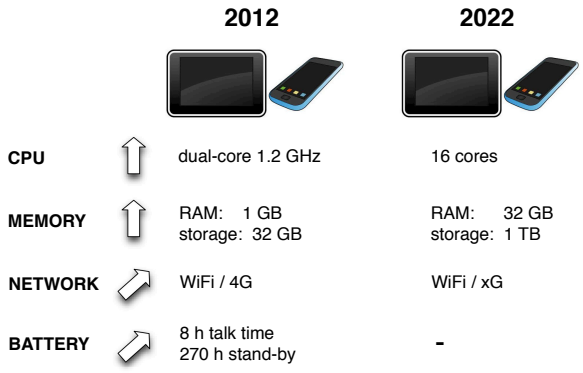

Figure 1: Mobile devices today and in ten years from now. While computation and memory will likely increase considerably, network bandwidth and battery capacity will not grow at the same pace.

If we then look ten years into the future, how would mobile cloud computing be different from now? Or, putting it differently, how would we redefine it given the enormous capabilities that our future mobile devices will expose? It is guaranteed that the answers to these questions are not straightforward, nor have a single solution. But at the same time, it wouldn't be wise for future mobile cloud computing architects to continue to look at these mobile devices as only capable of thin-layer interactions with remote clouds and not take into account the new exciting opportunities from their extraordinarily powerful hardware support.

A schematic and approximate representation of the evolution of these devices' characteristics is shown in Figure 1. While computation and storage will most likely continue to increase according to Moore's law, it is not clear if the cellular communication channel and the battery capacity will present similar trends. Battery still remains a bottleneck, not being able to cope with larger screens, computation, and data exchange requirements of many applications, either coming with the device itself or from the market places. Even the cellular wireless bandwidth might not exhibit substantial upgrades in the short term and often a new wireless

MCS'12, June 25, 2012, Low Wood Bay, Lake District, UK.

technology soon experiences lack of capacity given the continuous traffic growth due to the unstoppable demand for new data-intensive applications. It is the case of applications that, for example, nowadays heavily rely on the cloud to accomplish their tasks – speech and image processing are examples. Moreover, service providers prefer to design applications that accomplish most of the audio, video, and image processing in the cloud in order to be able to collect (and own) large amounts of user data, a precious resource for building more accurate machine learning classification models, for example.

Relying heavily on the backend cloud means, however, paying a cost on the network, where a large fraction of its bandwidth is employed to deliver "overhead" data (sent to the cloud for remote processing for example), which might overall degrade the network performance. Can we offload some of this traffic from the network when possible? Can we rethink the architecture of mobile computing systems in order to reduce the amount of unnecessary traffic on the cellular wireless links by accomplishing tasks locally? The freed bandwidth would be made available to other services, improving network usability and customers' satisfaction. The growing capabilities of mobile devices certainly open a door towards this possibility.

This paper presents our vision of *mClouds*, a mobile cloud computing architecture that runs resource-intensive applications on collections of cooperating mobile devices. Section 2 describes the mClouds design, particularly the mechanisms for managing the distributed computing operations between mCloud members. Section 3 discusses the user incentive scheme for mClouds, an important driver for the adoption of this technology.

2. MCLOUDS

In this section, we present the mCloud design and the role of mClouds within the context of future mobile cloud computing system architectures. mClouds are introduced to:

- take advantage of the improving hardware capabilities on mobile devices;
- reduce the impact on the cellular data channel, hence reducing bandwidth usage.

In what follows, we highlight the mClouds design principles and the motivations driving them.

2.1 Do It Locally If You Can

Recent studies on smartphone data usage show that the launch of the latest mobile device generations has triggered an exponential growth of wireless bandwidth demand, while highlighting that "*Just 1% of all users now consume half of the entire downlink data*" [5]. As an example of this growth, wireless data traffic handled by AT&T's network went from 0.1 Petabytes in 2006 to 27.1 Petabytes in 2011 – a 27,000% increase in 5 years [1]. A major reason for this increase is that many applications and services involving some computation – in the speech, image, and audio domains for example – heavily rely on the cloud to complete their tasks. The mobile device is in fact often considered only as a thin-layer abstraction for collecting data (audio, pictures, etc.), sending it to the cloud for processing, and receiving the output of this processing back. It's not hard to identify a drawback to this approach: it can suffer from network scalability issues when millions of devices engage in backend communication for cloud processing. With the growing popularity of smart, mobile devices and services, this issue may have an even larger impact in the future.

While the preferable repository for long-lived content (e.g., personal files, backups, etc.) will probably remain the backend cloud due to its security, reliability, and availability advantages, much of the processing could be gradually transferred to the mobile devices as their hardware becomes increasingly capable of handling heavy computational loads. Mobile devices will at some point be as powerful as today's backend machines, and task processing will require little or no support from the backend. This hypothesis is already reality for some applications that rely entirely on local processing. It is the case of PocketSphinx, for example, a successful effort of a speech recognition engine implementation for mobile devices without backend interaction [9]. Conti et al. [7] also introduce the opportunistic computing idea as a means to move computation to the mobile devices.

It has also been shown that a 3G cellular data interface requires 3 to 5 times more energy than WiFi transmissions [8]. Sending data remotely then not only claims large shares of wireless bandwidth, but also takes its toll on mobile device energy. By promoting local processing, mClouds is intended to: i) syphon data off the cellular data channel, and ii) reduce the backend cloud complexity by pushing intelligence to the edge, i.e., onto the mobile devices.

Local data exchanges between mCloud members can occur over free high-bandwidth WiFi networks, both ad-hoc and multicast, or through emerging short-range radio technologies specifically designed for direct interaction between mobile devices [4]. Should any remote data be required to support a local processing task – such as machine learning models or specific datasets – the data could be transferred to a mobile device opportunistically via WiFi, during off-peak hours via the cellular network, and, possibly, when the device is charging.

2.2 MCloud Processing

The mobile devices forming an mCloud are named *mDevs*. We use the term *mTasks* to denote the processing tasks carried out by mDevs. An example of an mTask is a speech processing procedure after collecting a voice sample from the user, or an image processing algorithm on a picture taken with the mobile device. We envision mechanisms within the mCloud such that mTasks can be executed either on the single device itself or, when possible, by distributing smaller portions of an mTask among members of the mCloud to parallelize the processing and alleviate the burden on single devices. We thus identify two scenarios: individual and distributed processing.

Individual Processing. This is the case of an mTask that can be autonomously accomplished by a single device, without the intervention of external entities. Single device computation is possible for mTasks that do not require too many resources (CPU, RAM, battery) to complete. PocketSphinx [9] is an example of individual processing.

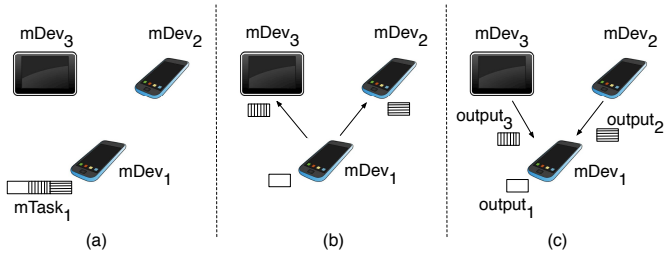

Figure 2: Distributed mCloud processing: (a) $mDev_1$ needs to accomplish $mTask_1$ and $mTask_1$ can be split into smaller subtasks; (b) master $mDev_1$ distributes the subtasks to the surrounding slaves; (c) master $mDev_1$ collects the results from the slaves.

Distributed Processing. There will be situations, however, where mDevs won't be able to act alone because the task requires too many resources. In such a scenario, mDevs could invoke the cooperation of nearby devices to alleviate its computational burden by slicing up a task into smaller subtasks for which independent execution is possible. We call the mobile device requesting the cooperation the *master* mDev and the others the *slave* mDevs. This scenario is depicted in Figure 2. If an mTask can be split into subtasks that can be safely executed independently from each other (as for $mTask_1$ from master $mDev_1$ in Figure 2a), the master can assign subtasks to those slaves willing to accept them.

Examples of mTasks that can be split into independent computation units are image and video processing operations, which can exploit high levels of parallelism. Each slave mDev independently proceeds to the processing of its assigned subtask and returns the result back to the master mDev. After collecting the results from the slaves, the master combines these results to produce the final output. Researchers have already proposed ideas to advance mobile computing, and mobile sensing in particular, to boost machine-learning inference accuracy by promoting inter-device cooperation [14]. Given the growing interest in mobile sensing applications to profile people's context and behavior, machine learning algorithms are among the main drivers for the mCloud technology: typical machine learning routines require heavy computation for which the support of powerful, backend machines is needed. With mClouds these tasks could be accomplished locally without needing to send large quantities of data to the backend cloud.

In order to support mCloud operations in any conditions, mDevs can always rely on the backend cloud for completing operations whenever it is not possible to successfully instantiate a distributed processing task in collaboration with nearby mDevs. This might occur because of the heterogeneous mCloud environment, where mDevs might have different capabilities that cannot fulfill the requests of the master. An mCloud is also a highly dynamic ecosystem, where members join and leave with little guarantees about the persistence of some of the resources. It might be also the case that nearby mDevs are not willing to accept a master's task.

2.3 MCloud Management

mClouds relies on two important principles. They are the ability to: i) easily manage the mCloud – its formation and functionality – which we discuss in this section, and ii) identify proper and effective incentive mechanisms for people to lend their devices for other people's computations (see Section 3).

An mCloud is an ensemble of mDevs that are usually mobile, dynamically joining and leaving the mCloud formation. Because of this, several questions arise: Would it be ever possible to form and maintain stable mCloud configurations? How do we know if the surrounding mDevs have enough resources to participate in a distributed processing task? What if there aren't enough mDevs in an mCloud to complete a task? This is only a small sample of the questions and challenges that need to be addressed. At a high level, we envision the following procedural steps:

Resource Discovery. An mDev (the master) that intends to instantiate a distributed processing task first needs to identify nearby mDevs that could potentially form the mCloud. At this stage, the master initiates a discovery phase by broadcasting solicitation messages, announcing the intent of forming an mCloud and the scope of the task. We only focus on one-hop master-slave interactions. Nearby mDevs willing to join the mCloud and take part in the cooperation, reply to the solicitation messages. If there are no nearby mDevs willing to join the mCloud, or there aren't enough mDevs to receive all the subtasks, the master relies on the backend cloud for the entire task or for the subtasks that cannot be accommodated by the slaves.

Formation. The mDevs (the slaves) willing to join respond to the master by sending their unique identifiers. The master maintains a list with the slaves' identifiers and the tasks assigned to them.

Maintenance. This is probably the most challenging phase, because due to mobility, the topology of the mCloud may vary over time (mDevs may move in and out of the master's short-range radio coverage). To this end, the resource discovery phase needs to be executed multiple times following the mCloud formation. The discovery could be scheduled to run periodically or, to reduce overhead and energy usage for message transmission, only in response to certain activity events. Researchers have shown how to run a user's activity classifiers on mobile devices using the motion sensors [12]. If then a slave detects the user's movement (e.g., walking), it might inform the master that it will be soon out of range and stop the processing. If the master detects its own movement, it would inform the slaves, which would react by halting their processes. Uncompleted tasks could be resumed on other mDevs or on the backend cloud.

Release. A release message, announcing the intention to leave the mCloud, can be sent by an mDev at anytime. Triggering factors are the assessment of lack of resources (e.g., the battery is depleting, the user launching a resource-intensive application on the device), mobility, or simply because the mDev doesn't intend to participate anymore without any particular reason. It is the master's responsibility to find other mDevs or involve the backend cloud for processing the tasks that haven't been completed by the released slaves.

Naturally, our design for mCloud topology discovery, formation, maintenance, and release can borrow techniques previously introduced and developed in other domains, such as mobile ad-hoc networks for example [6, 3].

3. INCENTIVES

Proper incentive policies are needed for users with mobile devices to opt-in to mCloud participation. In this section, we analyze possible incentive strategies that may help mClouds become a viable and effective alternative to the current mobile cloud computing model.

There has been recent work on managing cellular congestion by using economic incentives to modify the behavior of wireless users. Uninor [17] employs a dynamic pricing plan that gives different discounts based on the location and time of a voice call. TUBE [10] studies how time-shifting the wireless data demand helps the service provider by dynamically varying the incentives. We are interested in finding out what incentives (and from whom) should be provided to mobile device users to encourage them to participate in mClouds—since each mDev may bear the cost of cellular wireless bandwidth usage and battery drainage to help complete portions of mTasks.

Let's assume that a smartphone user has an mTask that can be distributed to other mDevs. We first start with a simple model (a revision of a classic pricing model proposed by Mendelson [13]) with the following parameters:

$\boldsymbol{u}$ – utility of a mobile service, a measure of satisfaction perceived by the mobile user, which may be indirectly revealed by the price the user is "willing to pay";
$\boldsymbol{T}$ – time to complete the service without using mClouds;
$\boldsymbol{w}$ – cost of waiting per second, which reduces the satisfaction and is highly dependent on the application;
$\boldsymbol{p}$ – cellular usage payment to carrier without using mClouds.

A mobile user will consider a mobile service useful if:

$$u - w * T - p > 0 \tag{1}$$

That is, there is still surplus in the utility after deducting the cost of the waiting time and cellular payment. When the network is congested, the delay component $w * T$ becomes a significant factor and fewer users would be willing to use the network. For example, a user may consider the value of watching a 100MB YouTube video of highlights of the latest NBA game to be \$5, and the cost of waiting (before continuous streaming starts) to be \$0.25 per second. Assuming the wireless service plan to be \$25 for 2GB, the cellular payment (without dynamic pricing) would be \$1.25. Under this model, the user would have no desire to watch the video if the delay is longer than 15 seconds (waiting cost of \$3.75).

If mClouds is available, then the user will have the option to distribute its tasks to multiple smartphones to reduce the time to completion. Let's first consider the scenario where the carrier does not provide incentives—and the user with the master mDev will have to pay other slave mDevs for their support:

$\boldsymbol{t}$ – time to completion with mClouds, typically smaller than T because the tasks are distributed;
$\boldsymbol{m}$ – number of slave mDevs;
$\boldsymbol{s}$ – average incentive payment to each slave mDev; each slave may price the same subtask differently;
$\boldsymbol{q}$ – cellular usage payment from the master mDev to carrier when mClouds is used.

Note that each slave mDev may handle some of the cellular wireless communications on behalf of the master mDev, and in that case q is likely to be less than p (less data to send to the backend). The payment s is typically made by the master mDev, but the carrier may also decide to pay the incentive to reduce network congestions in certain areas in order to keep the service at an acceptable level. A mobile user with the master mDev will consider a mobile service with mClouds useful if there is surplus in utility after paying the slaves:

$$u - w * t - m * s - q > 0 \tag{2}$$

And mClouds provides real cost savings (compared to equation 1) only if:

$$w * T + p > w * t + m * s + q \tag{3}$$

That is, the added payment to the slaves should be lower than the savings achieved through reduced waiting time and reduced payment to the wireless carrier by using mClouds.

Note that if the carrier is willing to cover the cost to significantly reduce the network congestion, then the slave payment cost (q) from the master is reduced to 0 and mClouds is beneficial to the master as long as:

$$w * T + p > w * t + q \tag{4}$$

In other words, incentives provided by carriers—intended to reduce congestion in their networks—may significantly boost the adoption of mClouds.

Now we turn the attention to the slave mDev. Note that the benefit of participating in mClous (through the incentives received) must outweigh the cost of the battery drain and cellular wireless bandwidth usage incurred by the slave. If b_i is the battery drain cost and v_i is the cellular data cost (as perceived by the slave i), then the following must hold for an mDev receiving payment s_i to be interested in participating:

$$s_i > b_i + v_i \tag{5}$$

For example, a recent measurement we conducted shows that the use of the "personal hotspot" application on an iPhone 4 device to support a WiFi-only iPad device consumes roughly 200MB of data during one hour of usage. In this case, v_i would be \$2.50 on a 2GB/\$25 monthly plan. The battery on the iPhone 4 drops by 22% for one hour of usage serving as a hotspot. If the slave mDev$_i$ considers every 10% of battery drain to be worth \$1, then b_i would be \$2.20. So the incentive payment s_i to the slave must be higher than \$4.70 if the task it received consumes the same amount of cellular data and battery usage.

Whether the incentives are paid by the master mDev or the carrier, the carrier can serve as the clearinghouse that handles all the transactions (or micro payments) on behalf of the owners of the mobile device participating in mClouds.

To help explain how incentives work in a real application scenario, let's consider an mCloud-based "real-time conference attendance" app, which allows a mobile user to take a snapshot of a crowd (say attendees in a particular session) and match faces in the crowd against the stored photos of all conference attendees to figure out who is attending which session. Furthermore, the conference organizers want to get real-time updates on the distribution of attendees in various sessions for logistics planning.

A conference volunteer may decide to use mClouds to quickly recognize all faces in the room since her own device may not have sufficient computation power and communication bandwidth to do that in real time. Let's assume that

there are 1000 attendees at the conference and a centralized database of all their faces is stored in the backend cloud. Let's assume that the app allows a user to upload a small image of a face to the backend cloud, which returns the name of the conference attendee. We also assume that there are typically 50 attendees in each session and 10 of them are available to participate in mClouds.

If the volunteer (with the master mDev) does it on her own device, the on-device app may take the following steps: (a) detect the number of faces in the snapshot and crop a small image for each face, (b) submit each face separately for recognition, and (c) return the list of attendees to the user. Let's assume that step (a) takes 10 seconds, step (b) takes 5 seconds for each face, and step (c) takes 2 seconds. The volunteer would need 10+5*50+2=262 seconds (T) without mClouds to accomplish the task. By using mClouds with 10 slave mDevs (each performing 5 face recognition tasks), the time will be reduced significantly since step (b) can be distributed to 10 mDevs. If we assume that the overhead of communicating with all slave mDevs is 2 seconds (through multicast), then the total time (t) with mClouds would be 10+2+5*5+2=39 seconds.

Now let's consider the surplus utility of the master mDev without mClouds. Let's assume that the utility of real-time face recognition (u) of 50 users in a conference session is worth \$10 to the conference organizers, and it costs \$0.03 in cellular payment for any phone to submit a face recognition request to the cloud, and \$0.02 is the cost of delay per second (w). Then $w * T$ would be \$0.02 * 262 =\$5.24 and p would be \$0.03 * 50 = \$1.50. So the surplus in utility (see equation 1) after subtracting the waiting cost and cellular payment would be \$10 -\$5.24 - \$1.50 = \$3.26.

Now we consider the case with mClouds. Let's assume that the average cost equivalent to the battery drainage b_i for performing the subtask on each slave mDev_i is \$0.10. The average cellular payment is simply \$0.03*5=\$0.15 each. The master mDev may decide to pay \$0.3 (> \$0.25) to each slave mDev to cover their costs. So the surplus utility for the master mDev (see equation 2) is \$10 - \$0.02*39 - 10*\$0.3 - 0 = \$6.22. Note that there would be no cellular payment on the master mDev since the tasks have been distributed. The use of mClouds would result in a saving of \$2.96.

If we further assume that each slave mDev is powerful enough to perform face recognition on its own, then all the cellular payments can be removed; however, each b_i would become higher since each slave mDev_i would have to work harder to finish its subtask without contacting the cloud. Note that the face recognition application on each slave mDev_i will only need a face classification model, with a much smaller memory footprint than the training data, to perform its task.

4. DISCUSSION

This paper takes the position that mClouds will be a core component of our future computing landscape. As we have described, running processing and storage-intensive applications on increasingly resourceful mobile devices would reduce demands on rapidly saturating cellular data networks.

It is important to note that mClouds will complement, but not replace, other approaches in the spectrum of mobile cloud computing solutions. In particular, we see two other approaches that will also play important roles. One is the currently dominant model of offloading computation and storage from mobile devices to centralized cloud resources. There are many scenarios in which this approach will continue to make sense, for example applications that need access to datasets too large to move away from centralized data centers. The other approach is Cloudlets, which shifts computation and storage to stationary resources close to the edge of the network [15]. This approach promises to reduce latency with respect to the centralized solution, and makes available more computing, storage, and energy resources than are available on mobile devices.

It is also important to mention that proper security and trust mechanisms are needed to facilitate the adoption of mClouds. We assume that mClouds are a trusted platform running on trusted devices where any third-party computation is executed within well-defined and secure environments [16]. Any phone-to-phone interaction should be regulated by off-the-shelf authentication and authorization techniques.

Finally, local stationary devices – such as personal computers, set-top boxes, etc. – could also become members of an mCloud. They could lend further computing power and enrich the capabilities of the mCloud.

5. CONCLUSION

We have presented mClouds, our vision of mobile cloud computing architectures, where mobile devices become core computing nodes because of their rapidly growing capabilities. We have discussed the advantages of the mClouds approach and provided initial design considerations. Although these ideas are preliminary, we believe that mClouds will play a key role in the future mobile computing landscape.

6. REFERENCES

[1] AT&T 2011 Annual Report. http://www.att.com/Common/about_us/files/pdf/ar2011_annual_report.pdf.

[2] AT&T Watson. http://tinyurl.com/7qvd4l4.

[3] Personal Networks. http://nrlweb.cs.ucla.edu/project/show/3.

[4] Qualcomm's Flashlinq. http://tinyurl.com/c3ba46s.

[5] Trends in Smartphone Data Use. http://www.arieso.com/news-article.html?id=89.

[6] S. Basagni, et al. *Mobile ad hoc Networking*. Wiley-IEEE Press, 2004.

[7] M. Conti and M. Kumar. Opportunities in Opportunistic Computing. *Computer*, 43(1):42–50, 2010.

[8] E. Cuervo, et al. Maui: Making Smartphones Last Longer with Code Offload. In *Proc. of MobiSys'10*.

[9] D. Huggins-Daines, et al. PocketSphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In *Proc. of ICASSP 2006*.

[10] C. Joe-Wong, et al. Time-Dependent Broadband Pricing: Feasibility and Benefits. In *Proc. of ICDCS'11*.

[11] E. Koukoumidis, et al. Pocket Cloudlets. *ACM SIGARCH Computer Architecture News*, 39(1):171–184, 2011.

[12] N. Lane, et al. A Survey of Mobile Phone Sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[13] H. Mendelson. Pricing Computer Services: Queuing Effects. *Communications of the ACM*, 1985.

[14] E. Miluzzo, et al. Darwin Phones: the Evolution of Sensing and Inference on Mobile Phones. In *Proc. of MobiSys'10*.

[15] M. Satyanarayanan, et al. The Case for VM-Based Cloudlets in Mobile Computing. *Pervasive Computing, IEEE*, 8(4):14–23, 2009.

[16] L.P. Cox and P.M. Chen. Pocket Hypervisors: Opportunities and Challenges. *ACM HotMobile'07*, 46–50, 2007.

[17] Uninor. Dynamic Pricing Plan with per Second Billing, Feb. 2010. http://bit.ly/oBl9kW.