Concurrent Access to Video Cameras in Edge-Centric Internet of Things

Nikita A. Bazhenov, Dmitry G. Korzun
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{bazhenov, dkorzun}@cs.karelia.ru

***Abstract*—Emerging Edge-Centric Internet of Things (IoT) environments become rich of video capture resources, including fixed cameras physically embedded into the environment (e.g., pantiltzoom (PTZ) camera), personal cameras carried by mobile users (e.g., smartphone cameras), and some other everyday cameras (e.g., cheap Internet Protocol (IP) cameras). This work presents a smart space-based solution to sharing the access information and control for video capture resources. Our demo service supports concurrent connection of one or more users (clients) to some IP-camera available in the IoT-environment. The solution implements two methods for connecting to cameras.**

There are several approaches to make connections to video cameras in modern edge-centric Internet of Things (IoT) environments [1]. A typical requirement is camera access to the Internet. Nevertheless, in some home environments or just using a standalone computer, the cameras are embedded (e.g., a laptop webcam) or connected via USB, and no Internet connection is needed to access the camera. The latter case is oriented to "one person uses the camera", as it happens in digital healthcare applications [2]. No concurrent access by other users and applications is possible.

In some other IoT environments, cameras can be more "public" [3]. Many users and applications (clients of the video-capture service) need connection to the camera to concurrently access and use the service. In particular, various IP-cameras provide such a service in many public areas, and connections from clients lead to concurrent access collisions.

We consider two methods for connecting several clients to a given camera. The first method is quantum control of time (see Fig. 1). The client sends a request, and if the camera is available (status 'Available') then the connection starts. A time quantum (e.g., 1 min) is allocated during which the client uses the camera service. Every new requesting client is allocated a time quantum and the queue is organized. When the client time elapses the next client has the access. The drawback is high delay for the service.

For example, one client requests to rotate the camera at the very last moment of its control (e.g., when 59 s elapsed). The camera delay is 2 s, and the rotation is 61 s after the request. The next client is able to control the camera only 59 s (instead of 60 s). Therefore, the last delay (if it exceeds the allocated time) should be considered separately.

Consider the following example.

...

59.2 s: A request for rotation from the client.

59.4 s: The camera start turning.

61.4 s: The rotation is completed. (Here the access control system should determine the end of use at the moment 60.0 s and ignore requests from the current client. Also, the time after 60.0 s is delayed)

61.5 s: The access control is given to the next client (instead of 60.0 s).

Since this method suspects the notification of clients when they can control the camera, the recalculation is performed after every next client. Obviously, the waiting time can be high (1 min and more in our example).

Advantages and disadvantages of the first method are the following.

+ Control division into time slots, during which the client can send consecutive commands.
+ Each client knows (at least approximately) the operation period with the video camera.
− Not all requests from clients are resolved (e.g., in case of large delays).
− The set time (e.g., 1 min in our example) is constantly shifted if users send long commands in the last fractions of seconds of their control.

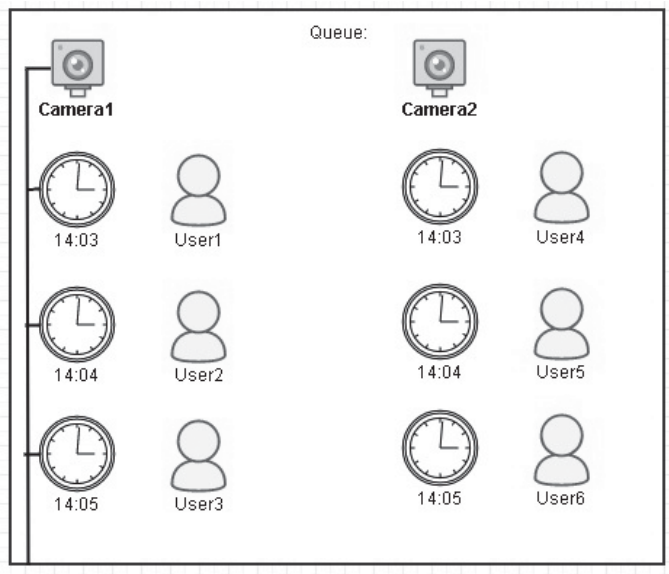

Fig. 1. Method 1 for accessing the camera: each client is allocated with a time quantum and all clients form a single input flow

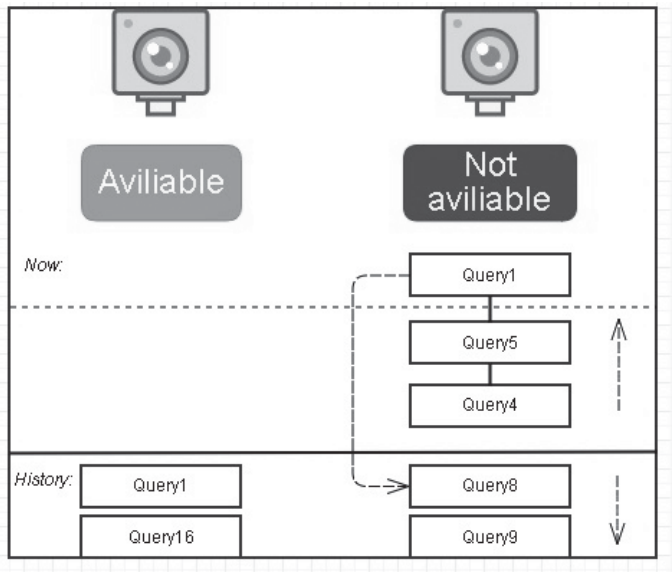

Fig. 2. Method 2 for accessing the camera: requests from many clients form a single input flow

The second method is shown in Fig. 1. The client (at any time) sends a request to the camera to make a particular operation. If the operation queue is empty at the moment, the request is executed instantly (with no excessive delay). If the operation queue keeps previous requests, then the request is pushed to the queue. In this case, collisions of concurrent access are simply avoided. When an operation is actually executed by the camera the client cannot control the execution. Moreover, the client can be already offline. In addition to the requests, the queue keeps some identities of the associated clients that request the operations.

If the client observes the status "Available" then the request is not necessarily be immediately performed. For example, two clients can simultaneously send requests to the same camera. The requests form a queue where the requested operations are performed sequentially.

This method should be supplemented with some restrictions on the concurrent activity of clients. For example, one user can not send more than one request per predefined time period (e.g., in 10 s). Even if the operation queue is empty, every client has to wait for some time before sending the next request. Also, the next request can be limited such that no new request before completion of the previous operation from the same client.

Advantages and disadvantages of the second method are the following.

+ No request is ignored by the camera; any incoming request is eventually resolved by the camera.
+ No excessive waiting for all preceding clients that occupy the camera.
− After sending a request it cannot be changed by the sender.
− The estimation of execution time for a sent request is complicated (for the case of many active clients).

The considered two methods can be generalized from the case "$n \geq 1$ clients and 1 camera" to the case "$n \geq 1$ clients and $m \geq 1$ cameras". For example, the client needs to rotate two cameras. Then, it is necessary to determine when the cameras are simultaneously free.

In this case, an intermediary is needed to take into account the whole access state. In particular, the access state describes which cameras are currently used by each client. To solve this task the smart spaces approach can be applied [4]. The access state is shared using a semantic information broker [5]. Both cameras and clients analyze this information for making access and request decisions.

ACKNOWLEDGMENT

The research is financially supported by the Ministry of Education and Science of Russia within project # 2.5124.2017/8.9 of the basic part of state research assignment for 2017–2019. The results are implemented by the Government Program of Flagship University Development for Petrozavodsk State University in 2017–2021.

REFERENCES

[1] A. Mohan, K. Gauen, Y. H. Lu, W. W. Li, and X. Chen, "Internet of video things in 2030: A world with many cameras," in *Proc. IEEE Int'l Symp. on Circuits and Systems (ISCAS 2017)*, May 2017, pp. 1–4.

[2] A. Meigal, K. Prokhorov, N. Bazhenov, L. Gerasimova-Meigal, and D. Korzun, "Towards a personal at-home lab for motion video tracking in patients with Parkinson's disease," in *Proc. 21st Conf. Open Innovations Association FRUCT*, Nov. 2017, pp. 231–237.

[3] S. S. N. Perala, I. Galanis, and I. Anagnostopoulos, "Fog computing and efficient resource management in the era of Internet-of-Video Things (IoVT)," in *Proc. IEEE Int'l Symp. on Circuits and Systems (ISCAS 2018)*, May 2018, pp. 1–5.

[4] D. Korzun, "On the smart spaces approach to semantic-driven design of service-oriented information systems," in *Proc. 12th Int'l Baltic Conf. on Databases and Information Systems (DB&IS 2016)*, G. A. et al., Ed. Springer International Publishing, Jul. 2016, pp. 181–195.

[5] S. Balandin and H. Waris, "Key properties in the development of smart spaces," in *Proc. 5th Int'l Conf. Universal Access in Human-Computer Interaction (UAHCI '09). Part II: Intelligent and Ubiquitous Interaction Environments, LNCS 5615*, C. Stephanidis, Ed. Springer-Verlag, Jul. 2009, pp. 3–12.