

EPS – Edge-hosted Personal Services for Mobile Users

Pengzhan Hao, Yongshu Bai, Xin Zhang, and Yifan Zhang

Department of Computer Science SUNY Binghamton Binghamton, NY

{phao3, ybai4, xzhang99, zhangy}@binghamton.edu

Motivation

More and more mobile workloads and optimizations now rely on the cloud (such as mobile cloud offloading, cloud-based web proxying and optimization, and cloud-based network traffic redundancy reduction). However, these cloud-based optimization may cause two problems:

- Long communication latency between data centers and mobile clients.
- Difficulty to perform large-scale and personalized support or optimization for mobile workloads without adding significant computation resource on the cloud.

Objectives

- O1: Enable developer-customizable and power-and-traffic-efficient network communication on the “last-hop” communication between mobile devices and the network edge, so that we can have
 - Better power/traffic efficiency for mobile device.
 - Better mobile service/optimization flexibility and scalability.
- O2: Distribute cloud services for mobile workloads to network edge, so that they can be done on a personalized basis, while enjoying much lower communication latency to/from mobile devices.

Our approach

Our approach is to deploy on the network edge certain computation endpoints (named as “service carriers”), each of which performs some specific service for individual users.

- Practical, secure, efficient and scalable framework.
- Support customized communication protocols between network edge and mobile devices

EPS architecture

Figure 4 shows the architecture of EPS.

- Edge node: access point, base station, etc.
- Service carriers (or EPS):
 - Each carrier only works with one mobile user
 - Each carrier only has one functionality
 - E.g. file sync optimization
 - Different service carriers can be run on the same node

In Figure 4, U1, U2 and U3 are in same last-hop wireless network. All these three users have their own service carriers. Like User 1, User 1 has func-1, func-2 and another EPS named EdgeCourier (an EPS-based system, details next).

Study 1: Fixing whole-file-trans. for cloud storage services

- For office suite documents, a small change always causes a substantial change at binary level
- Office suite docs are based on zip compression. Even we add just one character in a doc file, most cloud storage services transmit the whole updated file despite the fact that incremental sync mechanism is in place (as shown in Figure 1).
- Media type file used different encoding method
- Therefore, frequent small edits (which is common in mobile editing scenario) can cause repeated whole file transmissions.

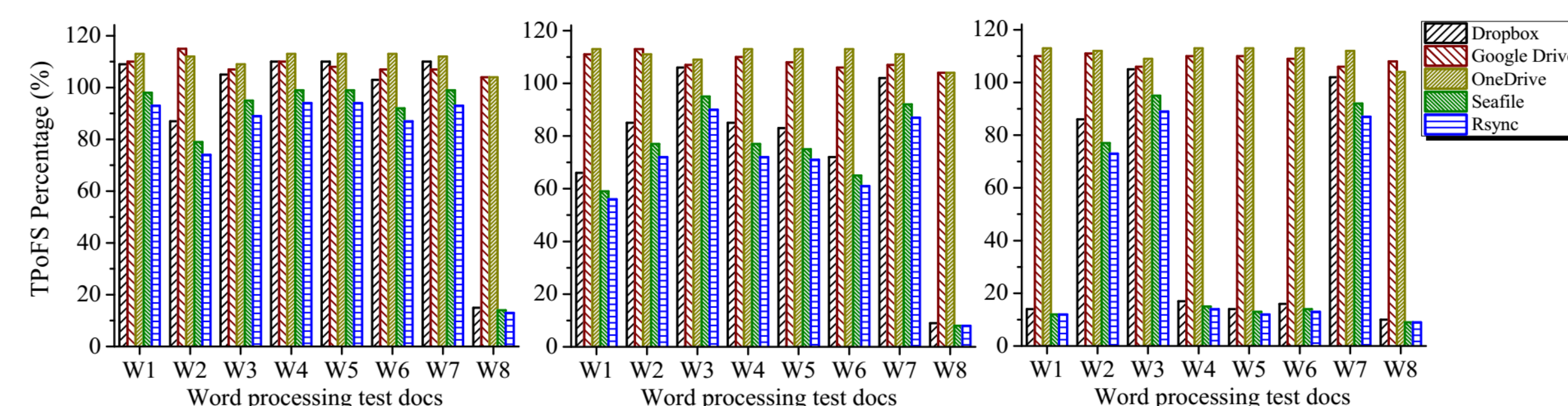


Figure 1. TPoFS performance of synchronizing a one-character-addition edit using three cloud storage services (Dropbox, Google Drive, and OneDrive) and two file synchronization software (seafile and rsync).

Study 2: Reducing redundant n/w traffic for mobile devices

Current web services adopt RESTful interfaces. Though the stateless property of the RESTful interfaces helps alleviate server load, it also introduces significant traffic redundancy on mobile devices.

- In many web services, HTTP headers takes the majority of each packet (e.g., as shown in Figure 2).
- In HTTP headers, most fields (e.g., cookies, user ID, etc.) are the same for all packets (but they need to be in every packet because of the stateless property).
- More than 80% (of header size) are redundant according to our measurement.

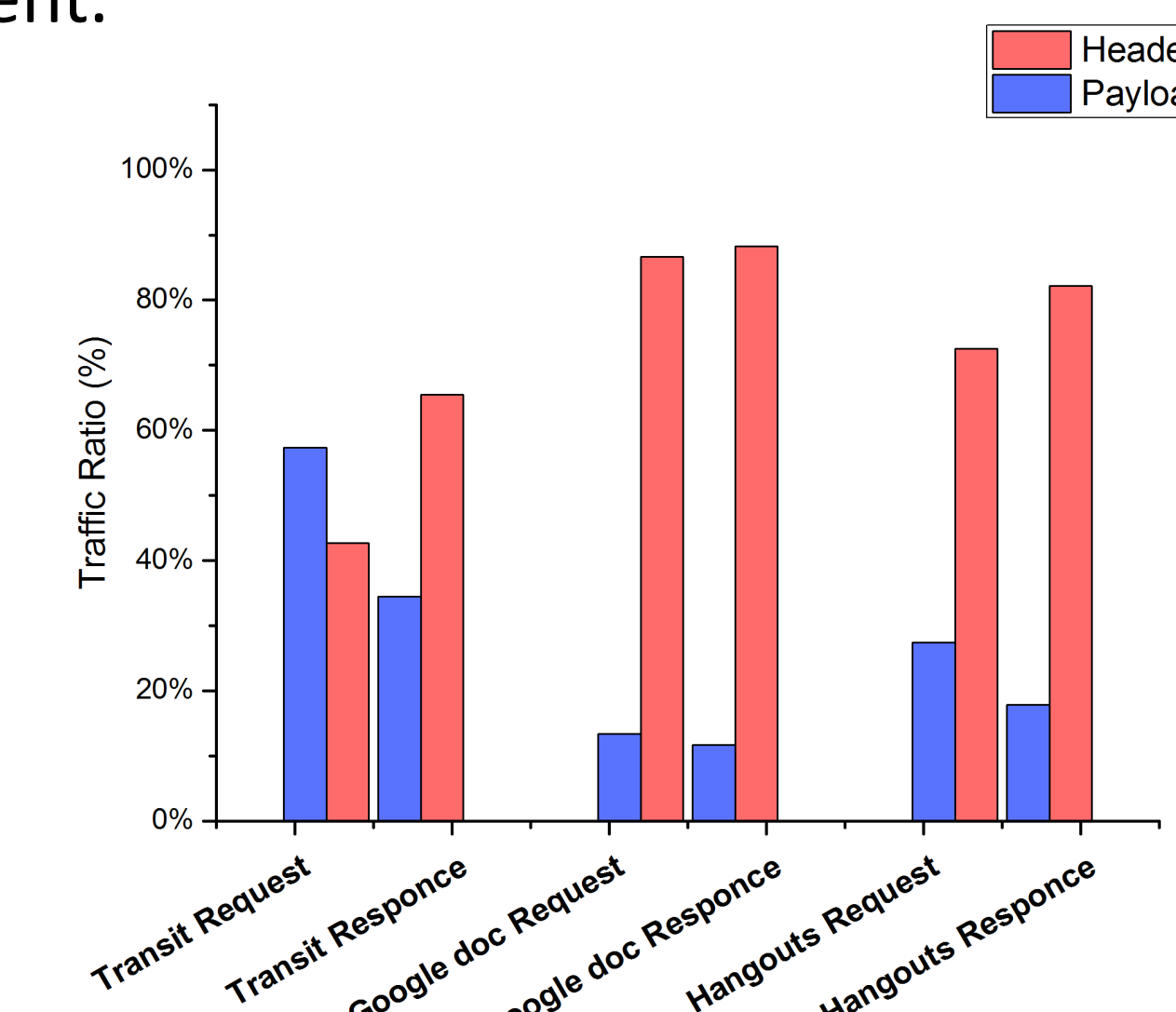


Figure 2. Payload and Header's ratio in whole traffic, including Google docs, transit and hangouts

Study 3: Personalized web improvement

Recent studies have shown that web experience improvement techniques (e.g., web caching) are dependent to mobile users' personal web browsing habits and preferences. Figure 3 shows how personal services deployed on the edge can help improve centralized cloud based centralized web services like Google's Flywheel.

- Low communication latency
- Personalized improvement
- Good flexibility and scalability

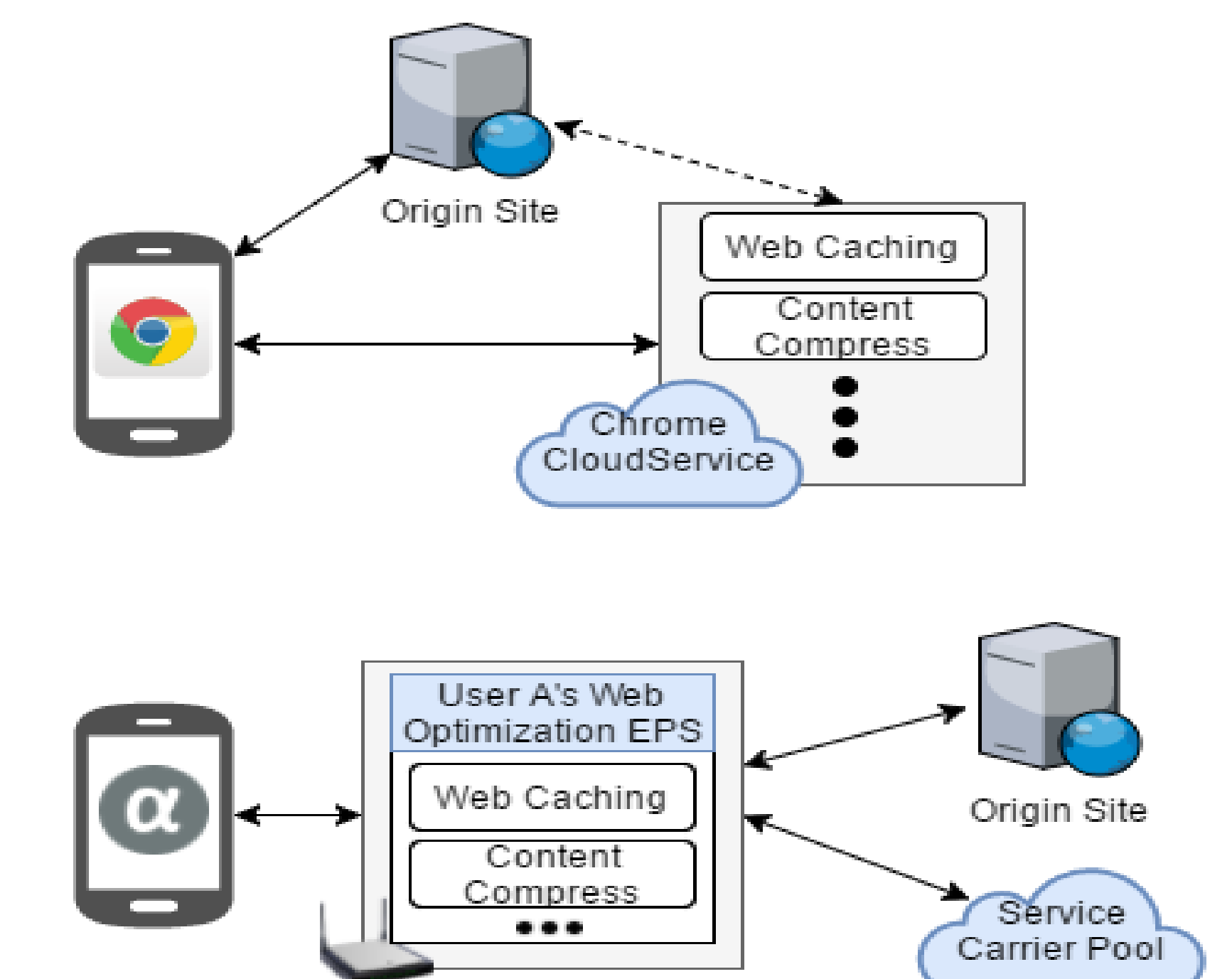


Figure 3: Google Flywheel's workflow and How it can benefit from Edge-based Personal Services.

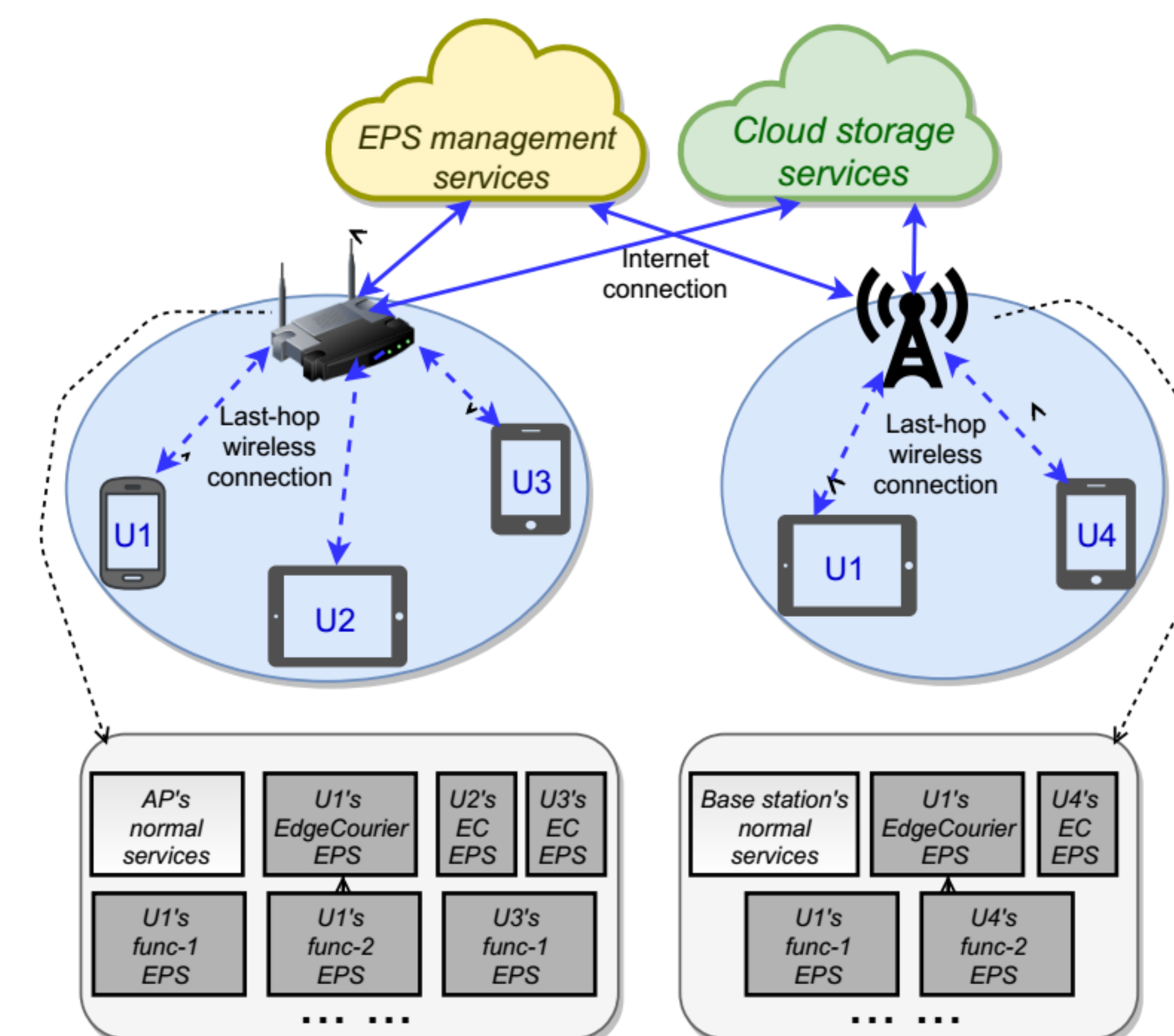


Figure 4: EPS Architectures

Research challenges

- Running tens and even hundreds of EPS instances efficiently in an edge node (e.g., an access point, which is usually resource-constrained) is challenging.

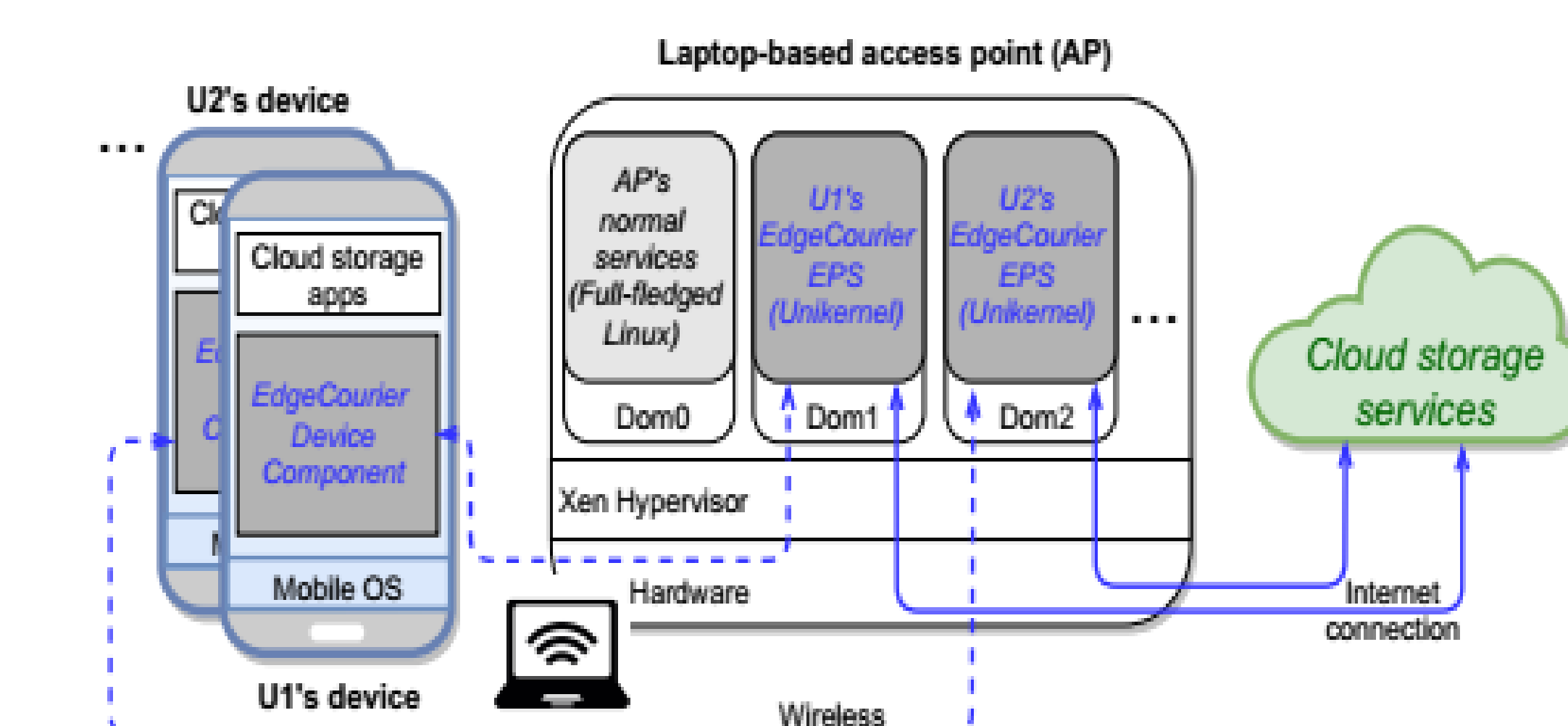


Figure 5: Edge-Courier service carrier's architecture. Edge-Courier is our solution for decreasing traffic when file syncing frequently but only with small changes.

- Resource sharing and communication between EPS instances while guaranteeing user privacy and security pose unique challenges.
- Efficient EPS instances management is challenging.

Ongoing and future work

- We are building several systems to demonstrate EPS's usefulness.
 - For example, we are working on an EPS-based system named EdgeCourier, solving the case-study-1 above. Figure 5 shows the architecture of our EdgeCourier system implemented with Unikernel used as the service

carrier. Table 1 shows how our EdgeCourier system can help in traffic reduction.

- We are now building our Xen-based Unikernel framework on ARM architecture.
- Starting from pure Xen mini-os, and porting the correct libraries.
- Automating Unikernel building process.
- Optimizing Unikernels so that they can be easily scaled in resource-constrained edge nodes.
- We will work on design and develop an EPS deployment framework for 3rd-party developers to easily make use of EPS's capabilities.
- We will work on the cloud-based EPS management services
 - Efficient service start/end/migration
- May borrow experiences from existing cloud/VM research. New challenges will come due to different applications and user experience requirements ..

Table 1: Traffic Reduction of Edge-Courier service carrier

Text size	Edit Loc.	File size after editing	Traffic (direct sync)	Traffic (sync with EC)
1K	Start	4,950	8,621	1,538
	Mid.	4,968	8,640	1,538
	End	4,950	8,618	1,538
100K	Start	79,874	83,551	1,538
	Mid.	79,912	83,587	1,538
	End	79,874	83,552	1,538
1000K	Start	760,214	763,894	1,538
	Mid.	760,215	763,894	1,538
	End	760,210	763,887	1,538