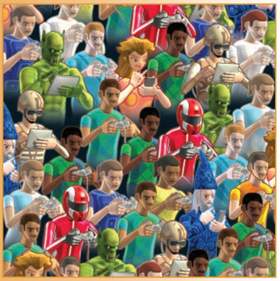

Toward Gaming as a Service

Gaming as a service (GaaS) is a future trend in the game industry. The authors survey existing platforms that provide cloud gaming services and classify them into three architectural frameworks to analyze their pros and cons and identify research directions. They also examine the features of different game genres to determine their impact on systematic design for cloud gaming services. Finally, they provide a vision on GaaS provisioning for mobile devices.

Wei Cai
The University of British Columbia

Min Chen
Huazhong University of Science and Technology

Victor C.M. Leung
The University of British Columbia

Cloud computing technologies and the computer game industry have made considerable advances recently. Thus, an active research topic is how to transform traditional gaming software and online games into *gaming as a service* (GaaS).[1] Similar to many cloud computing services, cloud gaming services have many advantages compared to traditional software systems, such as scalability for overcoming terminal hardware constraints, ubiquitous and cross-platform support for providing an immersive gaming experience, and cost-effectiveness for system development and software distribution. Moreover, the GaaS model exhibits many attractive features:

- *Effective antipiracy solution*. Because binary code is hosted in the secured cloud server, the GaaS model is a potential solution to the long-time and troubling problems of gaming software piracy.
- *Flexible business model*. Transforming from gaming software retailing to game service provisioning brings in more attractive and flexible support for many business models, such as pay-per-play, pre- and postpaid, or monthly subscription.
- *Click and play*. Game copies need not be installed in game terminals, which reduces the time cost for players. Gaming program size has increased significantly, but many games are rarely played after installation. Therefore, click and play can attract more potential players' attention, increasing a game's popularity.

Here, we investigate the essence of cloud gaming design from the software program perspective and survey various architectural frameworks for GaaS provisioning, analyze their benefits, and identify future research issues.

Essence of Cloud Games

Essentially, a computer game is a software program written in some programming language. Regardless of object- or procedure-oriented design, we consider gaming as a loop procedure that enables interaction between players and game logic. This particular procedure might contain different I/O methods and involve information exchange between multiple players. However, in general, we can view a game program as a series of interconnected modules with distinct functionalities.

Figure 1 illustrates the structure of an online game consisting of four main modules: the *input module* receives control information from the player; the *game logic module* is in charge of manipulating gaming content; the *networking module* exchanges various information with the game server, including how the avatars interact with others; and the *rendering module* renders the game video and presents it to the player. Looking at the game logic module's details, we can further divide it into several components that invoke each other and interact with the network interface, rendering engine, and I/O interface to facilitate the gaming procedure. The red arrows in the figure demonstrate an interaction between the player and the game system during a gaming session. The input module relays the player's instructions to component 5. Afterward, the processed information is delivered to component 7, which invokes the networking module to conduct information exchange. After successive processing by components 6 and 3, the rendering module generates and transmits the gaming video to the player's screen.

So, what is the essence of a cloud-based game from this perspective? The three graph cuts in Figure 1 illustrate the answer. Cut 1 demonstrates all modules being implemented at the terminal, while the networking module is the interface between game clients and the online gaming server. In this case, the cloud is used only as an information exchange server, which is the traditional design for online games. Cut 2 goes to the other end of the spectrum: the terminal contains only the input module, whereas the cloud hosts all of the remaining modules and components. In this case, the rendered real-time gaming video will be transmitted to the player via the Internet. We name this model a *remote rendering GaaS* (RR-GaaS). Cut 3 illustrates another design idea: the input and rendering modules are executed in the terminal, while the other modules run in the cloud. We denote this approach as a *local rendering GaaS* (LR-GaaS). According to these graphic cuts, we can see that cloud gaming's essence is to leverage cloud resources to execute several gaming modules, thereby reducing terminal workload and increasing efficiency.

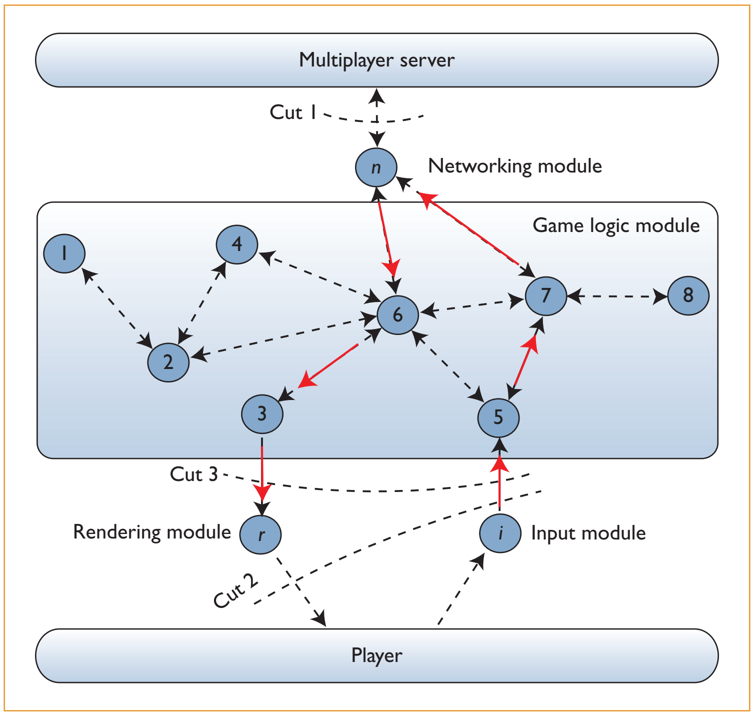

Figure 1. A modularized cloud gaming program. Red arrows demonstrate an interaction between a player and the game system during a gaming session.

Remote Rendering GaaS

RR-GaaS is the most mature cloud gaming service model. Companies such as OnLive, Gaikai, and G-Cluster are providing commercialized GaaS to the public, following a business model of executing source code on a gaming platform operated by a cloud service provider or network operator and delivering video frames to users' devices through portals.[2]

Architecture

Figure 2 shows the architectural framework for RR-GaaS. In this model, the cloud virtualizes an execution environment and initiates a game instance once it receives a connection request from a player. Depicted as *data flow*, the game instance renders the real-time gaming video in the cloud, after which the video capturer records it frame by frame. Then, the cloud gaming server transmits the video frames to the player's terminal through the Internet, after they're encoded in the video encoder. These encoded frames are reconstructed at the video decoder in the game terminal, and the video player displays them on the player's screen. In reverse, as *control flow* illustrates, the user controller records the player's inputs and the terminal transmits them to the cloud server after encoding by the input encoder. The cloud server receives these encoded signals and

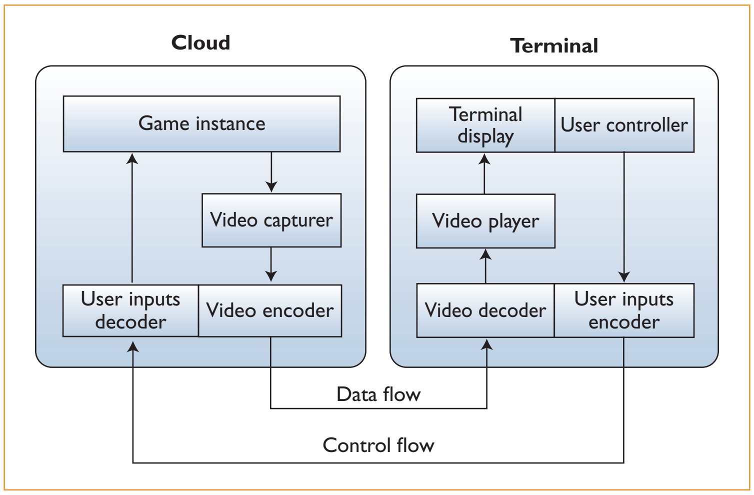

Figure 2. Remote rendering gaming-as-a-service (RR-GaaS) architecture. In this model, the cloud virtualizes an execution environment and initiates a game instance once it receives a connection request from a player.

decodes them into control inputs for the game instances to reproduce the player's interactions in the cloud.

System Benefits

In the RR-GaaS model, the gaming terminal becomes a thin client that intrinsically acts as input receiver, frame decoder, and video player. This implies that the hardware requirement for the gaming terminal is minimized, irrelevant of the complexities of game scenes, game logic, and interplayer interactions. Consequently, players can play high-end games with low-performance devices.

Challenges and Research Issues

The bottleneck of the RR-GaaS model is real-time video encoding and network transmissions. It is well known that high-quality and high-frame-rate video transmissions require a huge amount of bandwidth, whereas the gaming experience is highly sensitive to network latency (in general, the tolerable interaction delay is less than 120 ms). Consequently, the most critical research challenge in this area is establishing an effective real-time video encoding, compressing, transmission, and decoding system specifically designed for gaming.

Related Research

To address the real-time video transmission issue in the RR-GaaS model, improvements in rendering, encoding, and compression technologies are beneficial. Prior work introduces a video encoder that selects a set of key frames in the video sequence[3] and uses a 3D image-warping algorithm to interpolate other noncritical frames. To assist video coding, this approach exploits the pixel depth, rendering viewpoints, camera motion pattern, and even the auxiliary frames that don't actually exist in the video sequence.

Considerable work addresses the contradiction between the varying network environment and real-time video transmissions' high-bandwidth requirement. One system design monitors the round-trip time (RTT) jitter as the network end-to-end status indicator and adapts the video encoding parameters accordingly.[4] Another representative work summarizes all objective and subjective quality-of-service (QoS) factors that affect players' quality of experience (QoE) and formulates an impairment function to model the mobile game user experience (MGUE).[5] The authors propose a set of MGUE-oriented optimization techniques to address the challenges a wireless network faces in achieving acceptable response time for a gaming session.[6] The rendering parameters involve realistic effect, view distance, texture detail, environment detail, and rendering frame rate. Another work presents GamingAnywhere, the first open cloud gaming system.[7] Extensive experiments show that it provides higher responsiveness and better video quality than two well-known cloud gaming systems, OnLive and StreamMyGame.

Local Rendering GaaS

With improvements in hardware performance, most gaming terminals, including mobile devices, can perform complicated rendering for game scenes. Under this circumstance, the LR-GaaS model might be attractive, given that the rendering module is implemented in the gaming terminal to eliminate the high burden of real-time video transmission on the network.

Architecture

Figure 3 depicts the architectural framework for LR-GaaS. As we can see from the data flow, the gaming video is no longer rendered in the cloud server. Instead, the gaming logic generates a set of display instructions to represent the gaming graphics and sends them to the gaming terminal via the Internet. The terminal then interprets the display instruction with a designated instruction set and renders the gaming video locally at the terminal. The reverse control flow is similar to that in the RR-GaaS model.

System Benefits

The most highlighted benefit for the LR-GaaS system is that the cloud server no longer needs to transmit real-time gaming video frames to the terminals through the Internet, which significantly reduces the network workload. Otherwise, LR-GaaS has similar benefits to RR-GaaS, including click-and-play, antipiracy, and development cost reduction. Furthermore, this model still supports a cross-platform gaming experience, given that corresponding implementations of video renderers for multiple platforms are available.

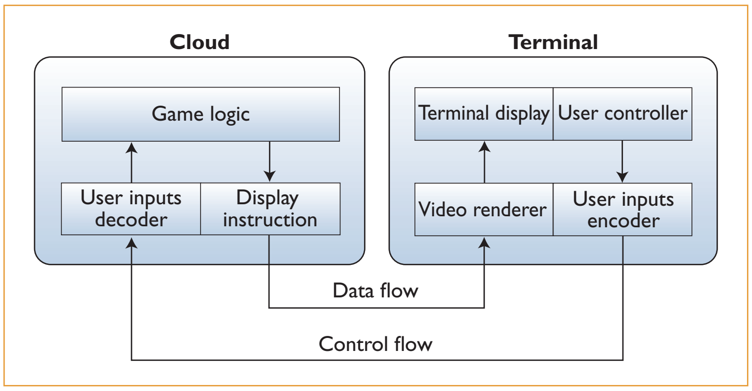

Figure 3. Local rendering gaming-as-a-service (LR-GaaS) architecture. The gaming video is no longer rendered in the cloud server. Instead, the gaming logic generates a set of display instructions to represent the gaming graphics and sends them to the gaming terminal via the Internet.

Challenges and Research Issues

For the LR-GaaS platform, the most critical challenge is to design an instruction set that can represent all gaming images for various games, and can be efficiently transmitted via the Internet. How to efficiently and accurately interpret the display instruction and render the gaming video in the terminal is also an open research issue.

Related Research

Browser games (also known as Web games), which follow the LR-GaaS model, have great commercial potential. As an indispensable application in computers and most mobile devices, the browser is one of the most pervasive thin clients. Moreover, HTML5 and the upcoming JavaScript libraries boost browsers' representation capacity, which makes them an ideal client for the LR-GaaS model. We can roughly categorize browser game evolution into three phases.

Script phase. The initial phase of browser games enables casual games that require lower image quality and less frequent interactions — for example, chess and card games. These games are usually simple and implemented by script languages that manipulate the static HTML elements with the Document Object Model (DOM).

Plug-in phase. The emergence of browser plug-ins such as Adobe Flash and Java Applet enriches the representation of browser games and provides a sophisticated integrated development environment (IDE) for designing and deploying large-scale video games on browsers. Accordingly, 3D-rendered games begin to appear in browsers.

HTML5 phase. Plug-in-enabled browser games still suffer from incompatibility among different platforms. With the recent, rapid development of the HTML5 standard and related Web technologies, plug-in-free browser games have become the new trend. In fact, plenty of HTML5 game engines — for example, Akihabara and ammo.js — already provide powerful code libraries to support the development of plug-in-free browser games.

Cognitive Resource Allocation GaaS

The intrinsic difference between RR-GaaS and LR-GaaS is the rendering module's execution environment, as graphic cuts 2 and 3 in Figure 1 indicate. This observation gives rise to the following questions: Is there a graphic cut that makes the system more efficient under different conditions? Is there a more flexible solution that adapts the cloud gaming service to varying circumstances, such as unstable network connectivity? To this end, we recently proposed a new design methodology[8]: constructing a cloud-based game using a set of interdependent components that execute in either the cloud server or the player's terminal, as determined by the terminal's current conditions and its network connectivity. In other words, cognitive capabilities that enable the cloud gaming platform to select optimal component combinations according to the system context and dispatch

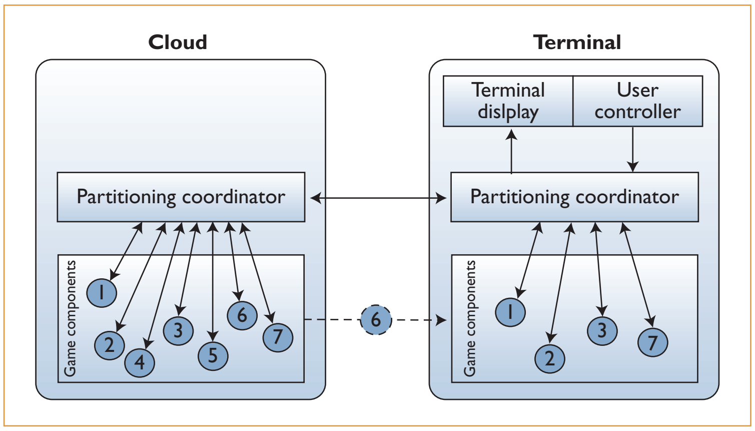

Figure 4. Cognitive resource allocation GaaS (CRA-GaaS) architecture. In this model, game programs are modularized as components, which can migrate from the cloud to the terminal during a gaming session and dynamically concatenate with each other to form a complete game.

components as mobile agents could provide a high-efficiency GaaS model.

Architecture

Figure 4 illustrates the architectural framework for the *cognitive resource allocation GaaS* (CRA-GaaS) model. In this category, game programs are modularized as components, which can migrate from the cloud to the terminal during a gaming session and dynamically concatenate with each other to form a complete game. That is, the terminal can fetch and execute a set of redundant game components from the cloud to reduce the cloud's burden. As the figure shows, a partitioning coordinator layer incorporating cognitive capabilities manages game component execution. Once the player conducts a control instruction, the coordinator intelligently assigns the responsive components, whether in the cloud or in the terminal. The same mechanism applies to the invocation between components: to achieve dynamic partitioning, the partitioner schedules all invoke messages.

System Benefits

As a flexible and intelligent platform, CRA-GaaS dynamically adapts its service to the component features and the varying system environment. In particular, the platform monitors the real-time environment status, sets an optimization target — for instance, the overall interaction latency, computational resource upper bound, or bandwidth ceiling — and achieves the target through dynamic partitioning. To some extent, RR-GaaS and LR-GaaS, which are based on a static partitioning of modules, are special cases of the CRA-GaaS model.

Challenges and Research Issues

As a cognitive system, achieving adaptive system optimization in real time is the most critical challenge for CRA-GaaS. Because a component can invoke another remote component based on the partitioning coordinator's scheduling, the cognitive platform explicitly controls the communication latency to satisfy an acceptable QoE level. In addition, because identical components might reside in both the cloud and terminal, developing an efficient mechanism to synchronize the data in the components is a challenge. To support click and play, no gaming component resides in the terminal at the beginning of the gaming session. Therefore, how to efficiently dispatch the necessary components to the terminal without interrupting the gaming experience is also an open issue.

Related Research

In our prior work, we first design and implement a cognitive gaming platform that supports both click and play and cognitive resource allocation.[9] Consisting of a set of controllers and coordinators, the cognitive platform can dispatch selected gaming components from the cloud to a player's terminal and later use dynamic partitioning to adapt its service QoS to the real-time system environment. As a development-friendly environment, the platform provides a set of APIs so that game developers need not be concerned with the lower-layer resource management details but can focus on game program design.

Selecting Provisioning Solutions

So, how can developers and service providers select from the three GaaS provisioning solutions? We answer this question from the perspectives of game genres and mobile terminals.

Game Genres

We classify game genres on the basis of two elements. *Scene observation* is how a player's observation of the game scene determines the variety of output images on the screen. In general, the most common scene observation styles are categorized as first-person, third-person, and omnipresent. *Game topic* refers to the game content provided, which determines the interaction behavior between players and the game. Game topics include shooting, fighting, sports, turn-based role-playing (RPG), action role-playing (ARPG), turn-based strategy, real-time strategy (RTS), and management.

Similar to previous work,[10] we give the relationships between the scene variety and motion frequency for the most common game genres, as Figure 5 shows. The figure indicates that these features vary among distinct game genres. Therefore, we must consider each architectural framework's pros and cons and use the best solution for the target gaming services. In general, games with high scene variety, such as omnipresent RTS and first-person shooting games, require a high bandwidth in video transmissions if the game provider chooses the RR-GaaS model. An LR-GaaS solution might thus be a better alternative. In contrast, RR-GaaS is a perfect solution for low-scene-variety games with high motion frequency, because the system can optimize the video throughput while supporting the massive and bursty intracloud information exchanges that occur among players. On the other hand, turn-based and management games tend to be more tolerant of overall latency than real-time games. Thus, such game genres will loosen the constraints on latency for the CRA-GaaS model, so that the system can further optimize its efficiency.

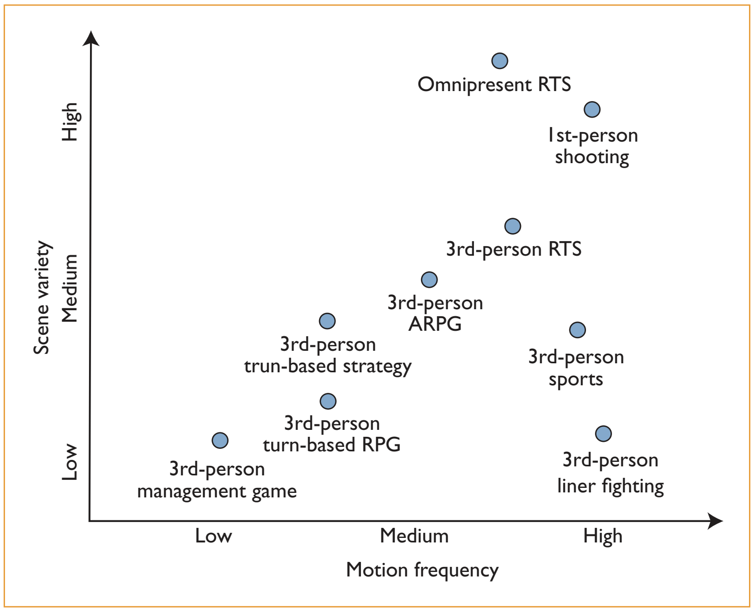

Figure 5. Scene variety and motion frequency. Such features vary among distinct game genres.

Mobile GaaS

Mobile games are among the most profitable applications in today's mobile market. Nevertheless, in addition to piracy problems, platform fragmentation — which increases development and deployment costs — also troubles game developers. Providing GaaS to mobile devices is a potential solution to address these critical issues.

However, the design and development of GaaS for mobile terminals also face many challenges. For example, RR-GaaS isn't a feasible solution for mobile devices, given that unstable and bandwidth-limited mobile networks might not be able to support real-time gaming video transmissions. Moreover, for most mobile network subscribers, receiving a large quantity of gaming video frames from a paid cellular Internet is unaffordable. In contrast, LR-GaaS provides a more practical solution. Nevertheless, it requires industry and academia to work together to further optimize mobile browsers' performance to improve not only image rendering but also the interactional support specifically designed for mobile games — for instance, fast-responsive touch gestures for gaming. Whereas RR-GaaS and LR-GaaS represent existing solutions, CRA-GaaS represents future solutions for mobile games. The mobile-agent-based dispatch solution could let mobile devices extend their gaming functionalities in a flexible fashion. Furthermore, with the dynamic integration of cloud resources, the platform can intelligently balance workloads for the cloud and mobile terminals with diverse capabilities to optimize the whole gaming system while satisfying players' QoE requirements.

GaaS provisioning is considered the next-generation modality of the gaming industry. To the best of our knowledge, ours is the first work to summarize all the platforms targeted toward providing cloud-based gaming services. We believe that in a very near future era of GaaS, players will be able to access their desired gaming contents ubiquitously through terminal devices with various capacities. To this end, we look forward to future improvements in CRA-GaaS regarding the study of the player-game interactive model, cognitive decision making, and unified human-computer interfaces for distinct terminal controllers.

References

1. P.E. Ross, "Cloud Computing's Killer App: Gaming," *IEEE Spectrum*, vol. 46, no. 3, 2009, pp. 14–14.
2. A. Ojala and P. Tyrvainen, "Developing Cloud Business Models: A Case Study on Cloud Gaming," *IEEE Software*, vol. 28, no. 4, 2011, pp. 42–47.
3. S. Shi et al., "Using Graphics Rendering Contexts to Enhance the Real-Time Video Coding for Mobile Cloud Gaming," *Proc. 9th ACM Int'l Conf. Multimedia*, 2011, pp. 103–112.
4. J. Laulajainen, T. Sutinen, and S. Jarvinen, "Experiments with QoS-Aware Gaming-on-Demand Service," *Proc. 20th Int'l Conf. Advanced Information Networking and Applications*, 2006, pp. 805–810.
5. S. Wang and S. Dey, "Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach," *Proc. IEEE Global Telecomm. Conf.*, 2009, pp. 1–7.
6. S. Wang and S. Dey, "Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming," *Proc. IEEE Global Telecomm. Conf.*, 2010, pp. 1–6.
7. C. Huang et al., "GamingAnywhere: An Open Cloud Gaming System," *Proc. 4th ACM Multimedia Systems Conf.*, 2013, pp. 36–47.
8. W. Cai, V.C.M. Leung, and M. Chen, "Next Generation Mobile Cloud Gaming," *Proc. IEEE 7th Int'l Symp. Service Oriented System Eng.*, 2013, pp. 551–560.
9. W. Cai et al., "A Cognitive Platform for Mobile Cloud Gaming," *Proc. 5th Cloud Computing Technology and Science Conf.*, 2013, pp. 72–79.
10. M. Claypool, "Motion and Scene Complexity for Streaming Video Games," *Proc. 4th Int'l Conf. Foundations of Digital Games*, 2009, pp. 34–41.

Wei Cai is a PhD candidate in the Department of Electrical and Computer Engineering at the University of British Columbia, Canada. His research areas include gaming as a service, mobile cloud computing, cognitive systems, and software engineering. Cai received an MSc in electrical engineering and computer science from Seoul National University, Korea. He is a student member of IEEE. Contact him at weicai@ece.ubc.ca.

Min Chen is a professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include the Internet of Things, e-healthcare, mobile cloud computing, and 5G wireless networks. Chen received a PhD in electrical engineering from the South China University of Technology. He's a senior member of IEEE. Contact him at minchen@ieee.org.

Victor C.M. Leung is a professor of electrical and computer engineering and holder of the TELUS Mobility Research Chair at the University of British Columbia, Canada. His research interests are in the broad areas of wireless networks and mobile systems. Leung received a PhD in electrical engineering from the University of British Columbia. He is a fellow of IEEE, the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. Contact him at vleung@ece.ubc.ca.