Using Storage to Increase Data Rate

KID: 20210103

The ever-increasing demand for fast communication links has been one of the major factors pushing innovation in communication engineering. The natural way to meet this demand is to bring in more physical resources to be leveraged by communication networks, such as increasing the number of fiber-optic cables available in the network, expanding the electromagnetic spectrum allocated for wireless data transmission, using multiple antennas at communicating devices, or even providing the electronic device the ability to send and receive data at the same time (this is called full-duplex radio). Each of these solutions increases the quantity of a specific raw resource which can be converted into an increase in data rate. For instance, doubling the number of antennas at the sender and the receiver doubles the number of signal dimensions, which essentially doubles the communication rate; similarly, using a full-duplex radio allows a device to talk and listen at the same time (instead of spending only half of the total available time on each of these two phases), and thereby again, doubling the data rate in an ideal setting.

A new parallel approach, that is compatible with other innovations (such as the ones mentioned above), is to exploit storage or memory space available at the receivers. Allow me to explain using an example. Imagine a digital movie library storing two movie files, call them A and B, and two users, call them U1 and U2, who have subscriptions to this library. For the sake of our argument, let's say U1 and U2 reside in the same locality, both of them typically watch movies in the evening, and have a limited amount of free space in their devices that can be used by the library's app. The space available in each device can store just one movie. The server knows that internet traffic is low during early mornings, and so, uses this opportunity to 'push' some data onto the devices during this off-peak duration. It does so in the following way. It splits each movie file into two halves: file A into A1 and A2, and file B into B1 and B2, and pushes A1 and B1 into user U1's memory and pushes A2 and B2 into user U2's storage space. The server does this non-trivial data placement since it does not know ahead of time which user will demand which movie that evening, and the server wants to be prepared for all possibilities. Now suppose in the evening U1 wants to stream movie A and U2 wants to stream movie B. When the users place this demand to the server, the latter simply does an exclusive OR of the files A1 and B2, denoted as A2 XOR B1, and sends this coded file to both users. Recall that U1 already has A1 and he can get the remaining half of the movie A by first receiving the coded file from the library and performing one more XOR with B1, that is, (A2 XOR B1) XOR B1. The second user can use the same coded file to retrieve the missing half of movie B as follows, (A2 XOR B1) XOR A2. Not only has the movie library served the two users simultaneously with a single transmission, but it has also done so in just half of the usual required time since the size of the coded file is only half the size of each movie file. We can check that the library could perform similar coding operations if the users had placed a different set of demands: for instance, if both the users wanted to watch movie A, the server can send the coded file A1 XOR A2.

The approach we just saw illustrates what is known in the information theory community as coded caching, a technique to exploit the space available at the clients and to transmit smartly designed coded files to increase the effective data rate during peak traffic hours. Since digital memory is getting cheaper, intelligent caching techniques can be used in next-generation networks to reduce the traffic strain placed on communication links.

Continued…

The application of such techniques to content distribution networks, and for wireless caching in 5G is being considered. These applications arrive with an assortment of interesting challenges, such as privacy leakage, reducing the number of fragments that a file needs to be broken into for cache placement (this is known as subpacketization), predicting the popularity of files, and using this prediction to optimize the data placement in user caches.

From a more theoretical viewpoint, coded caching is closely related to a general problem, known as index coding, that seeks to exploit side information at the receivers to improve data rate. Some of the research work from my group focuses on the following aspects of index coding: what if the communication link from the server to clients is noisy (which is usually the case in practice), what if the files cached at the clients are corrupted, and can we reduce the complexity of the operations performed at the clients? These problems are interesting mathematically, and they have strong relations to graph theory, matroid theory, coding theory, and information theory. From an engineering perspective, this problem is related to distributed data storage, distributed computing (MapReduce), and multi-user communication in wireless networks.

Dr. Lakshmi Prasad Natarajan

Assistant Professor

Departments:
Electrical Engineering
Artificial Intelligence