AN ATTRIBUTE SWITCHING MECHANISM FOR MOBILE NETWORKS

Chen Yu [1], Weiping Li [2]

[1,2]University of Science and Technology of China, Hefei, Anhui 230027, China
yc328@mail.ustc.edu.cn, wpli@ustc.edu.cn

Abstract: This paper presents a new mechanism for data transmission over mobile networks. The ultimate goal of designing the next generation mobile network is to have a single network that can support all applications with different data attributes. The proposed mechanism in this paper is inspired by the Mechanism Design Theory that is an emerging branch of game theory and has been used successfully in economics. The proposed attribute switching mechanism provides incentives for network applications to label their data attributes honestly and processes the data according to their attributes so that all applications achieve their best possible data transmission results without taking unnecessary network resources. Our simulation results have shown that the proposed approach performs better than other methods.

Keywords: Attribute switching; Mechanism design theory; Data transmission; Network design

1 Introduction

As the demand on mobile communication grows, there is a need to push the performance of mobile networks to achieve much greater throughput, much lower latency, and much higher reliability. However, optimization of the network usage to accommodate a wide range of applications plays an equally, if not more, important role in mobile network design. In the white paper for the fifth generation of mobile technology (5G) [1], it is envisioned to design the 5G architecture with "modular network functions that could be deployed and scaled on demand, to accommodate various use cases in an agile and cost efficient manner". For example, some network applications require extremely low latency and high reliability to transmit control data, other applications require high throughput and low latency to transmit visual communication data, and yet other applications require high reliability and extremely high throughput to transmit scientific experiment data. To efficiently use a single network for all these applications, the concept of "network slicing" is introduced in the 5G white paper, which means a flexible way to use a collection of 5G network functions and specific wireless settings that are combined together for specific use cases. However, in order to make network slicing work properly and achieve its intended objective, the network has to know the "true" attribute of the data to be transmitted.

There have been extensive efforts in network quality of service (QoS) area [2, 3, 4, 5], which tried to provide different levels of service for different applications. However, the key problem is that these approaches cannot avoid user's cheating behaviors without excessive cost, and this leads to their limited usage in practice. Therefore, the question is whether we can design a mechanism so that all applications honestly label their packets to claim the resources they need, but not more than what they need. Some researchers have considered this question and provided some solutions, such as the effort at Intel Labs which considers an incentive-oriented downlink scheduling method for wireless networks with real-time and non-real-time data flows [6].

Mechanism design theory is an emerging branch of game theory [7, 8]. While the traditional game theory studies how strategic and rational players behave and interact in a system with a given set of rules, mechanism design theory focuses on how to design such game rules that align those players' self-interests with their overall benefits. It studies how to maximize the overall benefit on condition that players are all rational and only consider their benefits. The Revelation Principal in mechanism design theory even tells us that we need only consider those mechanisms that make all players tell their true types. Mechanism design theory has demonstrated its success in economics and political science and we believe that it can be used for designing network too, since user behaviors in a network is similar.

In this paper, we introduce the mechanism design theory into network design and model network applications as players and network mechanism as game rules. Based this model, we map application attributes to player types and design an attribute switching mechanism for downlink scheduling in wireless network. Then we evaluate this mechanism to see if any cheating behavior would benefit any applications. We also compare our scheduling method with the straightforward first-in-first-out method and the method proposed in [6]. Although we have not been able to derive the optimal scheduling algorithm based on the mechanism design theory, such a framework for comparing different scheduling methods provides insights into how to design the optimal scheduling algorithm.

The rest of the paper is organized as follows. Section 2 describes the concept of "attribute" that indicates different requirements from different applications for network resources. Section 3 presents the model mapping the network design problem into a problem in the mechanism design theory. Section 4 describes a specific implementation based on the attribute switching model.

978-1-5090-1246-6/16/$31.00 ©2016 IEEE

Section 5 provides simulation results. Finally, Section 6 concludes the paper.

2 Attributes of network applications

Different network applications have different requirements on network resources, rather than so-called absolute "priority". For example, network control data are extremely sensitive to latency and require very high reliability, but usually with very low data rate; real-time video communication data are sensitive to latency and have a relatively high data, but may tolerate some data loss; and transfer of a large amount of scientific data needs as much channel capacity as available and needs to be reliable too, but may not be sensitive to latency. These observations inspire us to consider introducing some penalty into network transmission while providing the best possible benefit to the applications that really need it. We define three categories of attributes, namely, critical (CR), rea-time (RT), and bulk (BK) as follows:

- Critical (CR): CR attribute is for data with extreme sensitivity to latency and loss, but with very low data rate. Therefore, the transmission penalty for such data is to assign very low capacity to this attribute while providing extremely low latency and high reliability. The penalty does not hurt such data, but prevents other data from taking advantage of the extremely low latency and high reliability.
- Real-Time (RT): RT attribute is for real-time applications that need low latency and relatively high data rate, but may tolerate data loss to some extent. Therefore, the transmission penalty for such data is to introduce intentional data drop so that other applications will not try to take advantage of its relatively low latency and high data rate, while data drop for such data does not hurt intended applications too much.
- Bulk (BK): BK attribute is for applications that need a large amount of channel capacity and reliable transmission, but do not care too much about latency. Therefore, the penalty for such data is to introduce intentional jitter that does not hurt such applications too much, but discourage other applications to take advantage of its high data rate and reliability.

The characteristics of the three attributes are summarized in Table I.

Table I The characteristics of the three attributes

Attribute (A)	Volume (V)	Speed (S)	Reliability (R)
CR	Low	High	High
RT	High	High	Low
BK	High	Low	High

As shown in the table, there are three aspects of data transmission, namely, volume (higher means high data rate), speed (higher means low latency), and reliability (higher means less data loss). No application should have all three aspects to be high, but at most two of them being high. Our proposed attribute switching mechanism uses this fact in its delivery of network data.

3 Attribute switching model

In this section, we first briefly describe the mechanism design theory, mostly based on the contents in [7, 8], and then describe how we map the network design into the mechanism design theory.

3.1 A brief description of mechanism design theory

(1) Game Player
A finite group of players who interact in a game. Generally, we assume there are N players in a game.

(2) Type
Each player i has available to it some private information $\theta_i \in \Theta_i$ which is termed its type. Let $\theta = (\theta_1, \theta_2, \cdots, \theta_n)$ and $\Theta = \times_i \Theta_i$ where Θ is a probability space with a probability distribution function P.

(3) Strategy
Each player may lie to the designer by telling the false type. Let $\hat{\theta}_i \in \Theta_i$ be the strategy or "reported" type of application i and $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_n)$.

(4) Outcome
The outcome o is the game result from players' strategy, i.e, $o = f(\hat{\theta})$.

(5) Mechanism
Denote the set of outcomes O, a mechanism is a function from Θ to O, $f : \Theta \to O$. This is how designer obtain the outcome according to each player's strategy.

(6) Player's utility function
It is defined as $u_i : O \times \Theta_i \to \mathbf{R}$, i.e., $u_i(o, \theta_i)$ denotes the benefit that player i of type θ_i receives from an outcome o. If $u_i(o, \theta_i) > u_i(o', \theta_i)$, the player of type θ_i prefers outcome o.

(7) Utility function of mechanism designer
We should define a parameter to express whether the mechanism is good or not. However, the designer does not know each application's type, but only the probability distribution function of the type. So, we use the expected value of the sum of all players' utility functions $E_\theta(u_0(o, \theta))$ to express it, where $u_0(o, \theta) = \sum_{i=1}^{n} u_i(o, \theta_i)$.

(8) Problem definition
When the mechanism is given, players choose their strategies for their own best benefits. A dominant strategy is optimal for a player no matter what the other players do. A strategy $\hat{\theta}_i \in \Theta_i$ is dominant if $u_i\left(f(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_i\right) \geq u_i\left(f(\hat{\theta}'_i, \hat{\theta}_{-i}), \theta_i\right)$ for all $\hat{\theta}'$ and

$\hat{\theta}_{-i}$. There are two goals for the mechanism designer. One is to make all players tell their true types, i.e., design a mechanism so that truth-telling is a dominant strategy, meaning $\hat{\theta}_i = \theta_i$ for all i satisfies the definition of a dominant strategy. The other is to find a mechanism to maximize $E_\theta(u_0(o,\theta))$ among those truthful mechanisms.

3.2 Mapping of network design into mechanism design theory

(1) Game Player
Assuming n applications, this set is denoted as $N = \{1, 2, \cdots, n\}$.

(2) Type
In each application, we assume a single attribute to simplify the discussion, i.e., θ_i is either CR, RT, or BK.

(3) Strategy
Applications may try to cheat the network by telling false attributes. They can alter attributes for each packet no matter what their true attributes are. The motivation of the cheat behavior is to obtain more network resources such as higher bandwidth, lower latency, and/or higher reliability. However, they cannot cheat the time when they send the packets and the size of the packets, because the network can accurately know when packets come and how many bytes they have. Network node allocates resources per the attributes that applications tell to the network.

(4) Outcome
The outcome o is the result of scheduling. We denote the set of outcomes O.

(5) Mechanism
Network node allocates resources according to the "attribute", size, and time of data packets. Every application knows if packets are lost and also when the packets are transferred. This is the outcome of scheduling. Actually, scheduling is the "mechanism" in our model of attribute switching design.

(6) Application's utility function
Each application judges the quality of service according to the outcome given by the mechanism.

(7) Utility function of network designer
We use the sum of utility functions of all applications to express the utility function of the network designer, i.e., $u_0(o,\theta) = \sum_{i=1}^{n} u_i(o,\theta_i)$.

4 An implementation of attribute switching mechanism

We have three types of applications with attributes CR, RT, and BK respectively. We assume each CR packet is size 0.1KB, each RT packets is size 2KB, and each BK packets is size 12KB. Each application has 1000 packets. We assume CR application sends packets with data rate of 16KB/s, RT application sends packets with data rate of 80KB/s, and BK application sends packets with data rate of 10000KB/s. There are 100 applications in a game. The probability of CR application is 0.3, the probability of RT application is 0.5, and the probability of BK application is 0.2. The time of each application coming follows a Poisson process with parameter $\lambda = 10$. The strategy of each application is to claim which type it is. It can tell truth or not. For example, if a CR application claims it is an RT application, then all the packets will be labeled by RT, but they are actually CR packets.

We assume the total bandwidth is 10000KB/s, and we set 100 queues that each one is prepared for one application. If one application claims that it is a CR application (maybe it is actually not) and its queue is not empty, we give this queue bandwidth of 16KB/s. If one application claims that it is a RT application and its queue is not empty, we give this queue bandwidth of 80KB/s. Then all BK applications whose queues are not empty will divide the rest of the bandwidth equally. For example, if 5 CR applications' queues, 4 RT applications' queues and 3 BK applications' queues are not empty. Then each CR application will have bandwidth of 16KB/s, each RT application will have bandwidth of 80KB/s, and each BK application will have bandwidth of (10000-5*16-4*80)/3=3200KB/s. By doing this, RT and BK applications may not claim they are CR applications because the bandwidth is too small. We will count the number of queues which are not empty when there is a packet sent to the mechanism. After all packets are sent over, we will count the number of queues every 0.1 seconds.

Each queue will forward the packets simultaneously according to their bandwidth using Token Bucket. Importantly we introduce intended packet drop for RT queues and intended jitter for BK queues. For an RT queue, we will randomly choose packets to drop with probability of 0.1. Then CR application will not want to claim they are RT applications, because any drop of packets will have fair influence. BK application may not claim they are RT applications because dropping will incur re-transmission, which results in longer time to finish the transmission. For a BK queue, we will double the bandwidth for the first 0.1 seconds, and 0 bandwidth for the next 0.1 seconds, and double the bandwidth for the next 0.1 seconds and so on. By doing this, the RT and CR applications may not want to claim that they are BK applications because, if doing so, lots of packets cannot be sent to the destination on time which has a serious influence.

The utility functions for applications are defined as follows. For a CR application, if any packet is dropped or any packet cannot be sent within 0.01 second, it will be scored 0; otherwise it will be scored 100. For an RT application, let L denote the data amount of all packets that is dropped or cannot be sent within 0.04 second. Let A denote the data amount of the application. Then $score = 100 * (1 - L/A)$. For a BK application, Let T denote the total time to transmit all packets. Then

$score = 100 * (1 - D/T)$.

5 Simulation results

We have done five experiments. The first experiment is to compare the scores of CR application between telling the true type and telling the false type. The second experiment is to compare the scores of RT application and the third is to compare the scores of BK experiment. The fourth experiment is to compare Attribute Switching Mechanism with FIFO Mechanism. The fifth experiment is to compare Attribute Switching Mechanism with Incentive-oriented Downlink Scheduling Method presented in [6].

For the first experiment, we will play the game following process by Section 4 for 100 times. For each game, we record the scores of specific CR application under the circumstances that this application tells the true type and that this application cheats the network by labeling RT and BK application. The rest applications are always telling the true types. We can see the results in Figure 1. Because the yellow line and red line are coincide, so we can only see the red line. This figure tells us CR application will always tell true type. This is reasonable. If CR application claims it is RT application, it will have about 10% packet lost which will make it terrible. If CR application claims it is BK application, it will have jitter, which make some packets are sent too slowly.

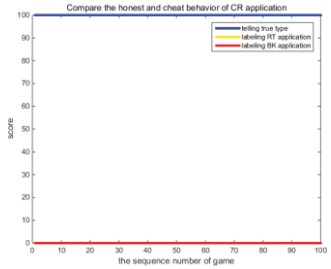

Figure 1 Compare with cheating for CR

For the second experiment, we repeat the process above but replace CR application with RT application. Figure 2 shows the result. This figure tells us RT application will always tell true type. If RT application claims it is CR application, the bandwidth will be too small, then packets cannot be sent on time. If RT application claims it is BK application, some packets will wait for so long time to be sent because of jitter.

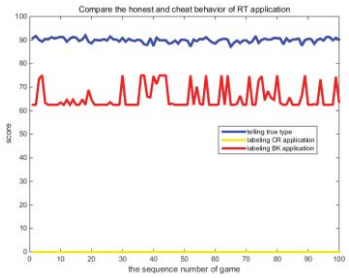

Figure 2 Compare with cheating for RT

For the third experiment, we repeat the process above but replace CR application with BK application. Figure 3 shows the result. This figure tells us BK application will always tell true type. If BK application claims it is CR or RT application, the bandwidth will be too small, then packets are sent over a longer time.

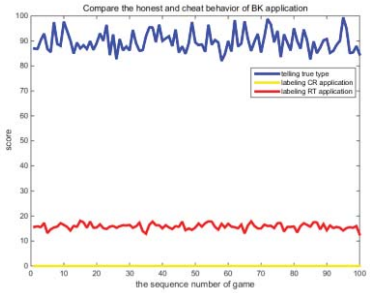

Figure 3 Compare with cheating for BK

For the fourth experiment, we also play games for 100 times. In each game, we record the mean score of all applications in the game using Attribute Switching Mechanism and using FIFO Mechanism. In other word, we compare the utility function of mechanism designer of two mechanism and see which mechanism is better. The result is shown in Figure 4. This figure tells us Attribute Switching Mechanism is better than FIFO Mechanism no matter the volume of buffer of FIFO Mechanism.

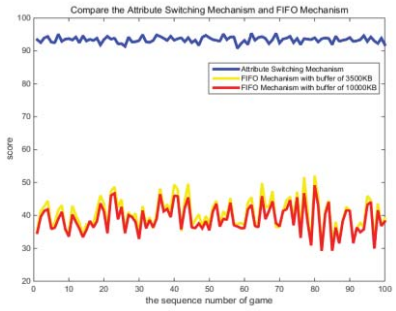

Figure 4 Compare with FIFO

This is reasonable because the only aspect that the FIFO Mechanism considers is channel capacity. As long as there is available capacity, it forwards incoming packets on a first-come-first-serve basis. It does not consider the effect of time delay and packet drop on the real-time application and control message. So the scores of RT and CR applications are extremely low, as shown in Figure 5 and Figure 6. For the BK applications, the performance of the FIFO mechanism depends on the packet arrival time. If the packets arrive in burst, jam of the previous packets affects the BK applications' scores. As shown in Figure 7, the FIFO mechanism for BK applications is not as good as the Attribute Switching Mechanism on the average. This explains why the average score of the FIFO mechanism is much lower in Figure 4.

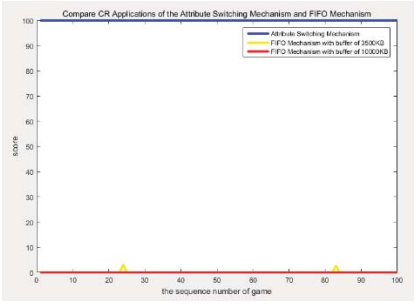

Figure 5 Compare with FIFO for CR

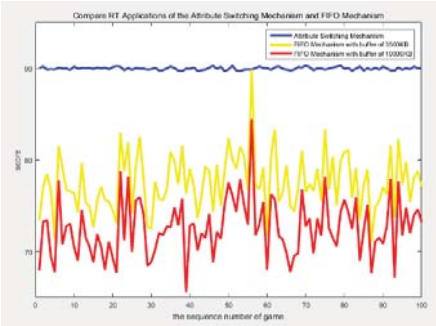

Figure 6 Compare with FIFO for RT

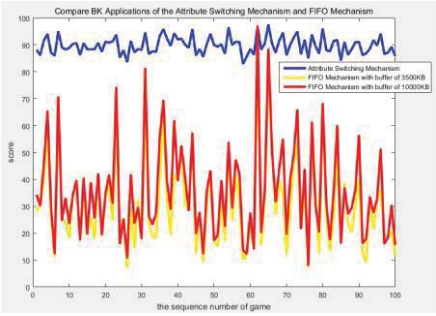

Figure 7 Compare with FIFO for BK

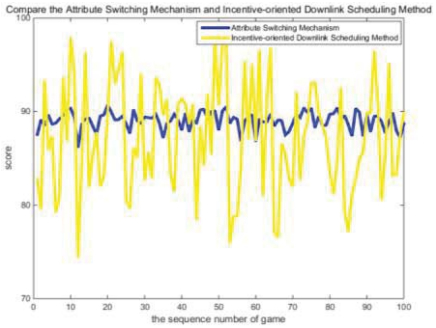

Figure 8 Compare with IDSM

In the last experiment, we compare Attribute Switching Mechanism and Incentive-oriented Downlink Scheduling Method (IDSM) in [6]. However, in Incentive-oriented Downlink Scheduling Method, there are only real-time (RT) and non-real-time (BK) applications, so we change some parameters in our game. In this experiment, there are only 70 applications. The probability of RT application is 5/7, and the probability of BK application is 2/7. Other conditions do not change. Then we repeat the process of the fourth experiment. The result is shown in Figure 8. The result shows that IDSM performance fluctuates in a wide range. After discussing with the authors of [6], we understand that IDSM may result in large packet drop for the RT applications due to enforcing the delay deadline (very short for the RT packets), if the BK packets come in burst to jam the output. Usually, there should be a rate control mechanism for the RT applications to throttle back the RT sending rate so that the RT packet drop is limited. Since the rate control at the source depends on the feedback about the packet drop situation from the network node, it is a little complicated to implement it. On the other hand, our mechanism does not need rate control and still provides stable performance.

6 Conclusions

We propose an attribute switching mechanism for data transmission in mobile networks, based on the mechanism design theory. Our simplified model of network design with simulation results have shown the advantage of considering network design as a mechanism design problem. Our initial results have shown that the scheduling algorithm performs well compared with other methods.

Acknowledgements

We would like to thank Dr. Jing Zhu and Dr. Rath Vannithamby at Intel Labs for the discussion on our work in comparison with IDSM. This work is partially supported by Intel Collaborative Research Institute.

References

[1] NGMN Aliance, NGMN 5G WHITE PAPER, 17 February 2015. http://ngmn.org/fileadmin/ngmn/content/images/news/ngmn_news/NGMN_5G_White_Paper_V1_0.pdf

[2] Alan Demers, Srinivasan Keshav, and Scott Shenker, “Analysis and simulation of a fair queueing algorithm,” Proc. SIGCOMM 1989, pp. 1-12, Sept. 1989.

[3] Ion Stoica, Hui Zhang, and T. S. Eugene NG, “A hierarchical fair service curve algorithm for link-sharing, real-time and priority services,” IEEE/ACE Trans. Networking, vol. 8, no. 2, pp. 185-199, Apr. 2000.

[4] Martin Karsten, “FIFO service with differentiated queueing,” Proc. ANCS 2011, pp. 206-211, Oct. 2011.

[5] Brian E. Carpenter and Kathleen Nichols, “Differentiated Services in the Internet,” Proc. IEEE, vol. 90, no. 9, pp. 1479-1494, Sept. 2002.

[6] I-Hong Hou, Jing Zhu and Rath Vannithamby, “Incentive-Oriented Downlink Scheduling for Wireless Networks with Real-Time and Non-Real-Time Flows,” 2013 IEEE Globecom Workshops, pp. 252-257, Dec. 2013.

[7] M.J. Osborne and A. Rubenstein, A Course in Game Theory. The MIT Press, 1994.

[8] Matthew O. Jackson. Mechanism theory. In The Encyclopedia of Life Support Systems. EOLSS Publishers, 200.