Towards Accurate Accounting of Cellular Data for TCP Retransmission

Younghwan Go, Denis Foo Kune[†], Shinae Woo, KyoungSoo Park, and Yongdae Kim

KAIST University of Massachusetts Amherst[†]

ABSTRACT

The current architecture supporting data services to mobile devices is built below the network layer (IP) and users receive the payload at the application layer. Between them is the transport layer that can cause data consumption inflation due to the retransmission mechanism that provides reliable delivery. In this paper, we examine the accounting policies of five large cellular ISPs in the U.S. and South Korea. We look at their policies regarding the transport layer reliability mechanism with TCP's retransmission and show that the current implementation of accounting policies either fails to meet the billing fairness or is vulnerable to charge evasions. Three of the ISPs surveyed charge for all IP packets regardless of retransmission, allowing attackers to inflate a victim's bill by intentionally retransmitting packets. The other two ISPs deduct the retransmitted amount from the user's bill thus allowing tunneling through TCP retransmissions. We show that a "free-riding" attack is viable with these ISPs and discuss some of the mitigation techniques.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection; C.2.1 [**Network Architecture and Design**]: Packet-switching networks; C.2.6 [**Internetworking**]: Standards

Keywords

Cellular Networks, TCP Retransmission, Accounting, Charging

1. INTRODUCTION

Cellular 3G/4G data traffic is rapidly increasing. The volume is predicted to reach 10.8 Exabytes per month in 2016, which is an 18-fold increase from that of 2011 [1]. The number of cellular network users has already reached 1.2 billion worldwide [2], and it is estimated that 85% of the world population will subscribe to the cellular network service by 2017 [3].

Given the increasing demand in the cellular traffic, accurate accounting of the traffic usage becomes all the more important. Most cellular ISPs adopt the pay-per-usage charging model for cellular Internet access. Subscribers typically buy a monthly usage plan (e.g., 3 GB per month) and the ISPs enforce it by byte-level accounting of the consumed IP packets. However, this approach presents an important policy decision for the TCP traffic. ISPs now need to decide whether they account for retransmitted TCP packets or not. If the ISPs reflect the retransmitted packets into the bill, it may be unfair to the users especially when the packet delay variance or losses are due to a poorly-provisioned infrastructure. In our measurement at one ISP in South Korea, we observed some flows with up to 93% of the packets being retransmitted due to packet loss. What is worse is that the blind accounting policy can be easily abused by malicious attackers that try to inflate the cellular traffic usage for a specific user or even for all users from a specific ISP. The natural alternative is to remove the retransmitted packets from the bill, but accounting becomes expensive since it has to manage every TCP flow for each subscriber.

In this work, we present the implications of byte-level accounting policies in the cellular traffic for TCP retransmission. The root cause of the problem lies in that the majority of the mobile data traffic flows over TCP [4–7], which ensures the flow-level reliability by transparently retransmitting the lost packets [8]. However, the ISPs account for each IP packet, which sometimes creates a disparity in what users perceive and what the infrastructure provides.

To better understand the current practice, we examine the accounting policies for TCP retransmission with five large cellular ISPs in the U.S. and in South Korea. Surprisingly, we find that the accounting policies vary between ISPs, and that even the ISPs in the same country have different policies. Our measurements reveal that three ISPs (two in the U.S. and one in South Korea) account for every packet regardless of TCP retransmission. We further confirm that the users in these ISPs can be the target of a usage-inflation attack that maliciously retransmits packets even if there are no packet loss. The remaining two ISPs (both in South Korea) intentionally remove the retransmitted amount from the usage statistics. However, we find that their implementation allows free data transfers if attackers tunnel their packets inside TCP retransmissions. This implies that the ISP accounting system checks only the TCP headers for retransmission and does not check the actual content of the payload; doing so could be expensive in terms of storage and computation to recall previous payload contents, and compare them to suspected retransmissions.

Our contributions in this paper are summarized as follows. First, we report that the current byte-level accounting for TCP retransmission fails to meet the fairness nor the correctness in billing. Blind accounting of every packet leads to unfair usage inflation if the retransmission happens due to infrastructure-induced congestion or degraded wireless links. Second, we show that the current practice of cellular traffic accounting is vulnerable to attacks that either inflate the usage or send the packets without being charged. Peng et.

ACM HotMobile'13, February 26–27, 2013, Jekyll Island, Georgia, USA.

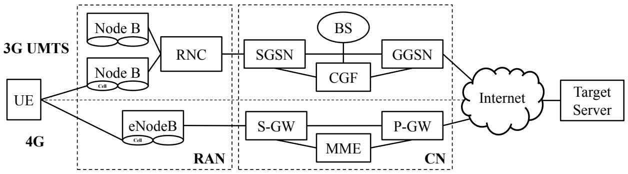


Figure 1: Overall architecture of 3G/4G cellular network

al. showed a similar attack exploiting a loophole in an ISP policy that blindly passes all packets on port 53 (DNS) at no charge [9,10]. While their work can be considered as "bugs" in the accounting policy, we believe that the accounting policy for TCP retransmissions is a *fundamental* problem tied to the basic mechanisms of the TCP layer. We argue that cellular ISPs should not count retransmitted bytes against the user's data plan, but they should also make sure that the retransmissions are legitimate to prevent abuse. Later in this paper, we discuss a few possible solutions that can be used to prevent (or mitigate) the vulnerability while maintaining a reasonable accounting load even for high-throughput networks.

2. BACKGROUND

In this section, we describe the basic architecture of 3G/4G cellular networks and their accounting process. We mainly focus on the Universal Mobile Telecommunications System (UMTS) [11] for 3G and Long Term Evolution (LTE) [12] for 4G. The architecture is based on a Packet-Switched (PS) domain, in which the data is transferred in packets [13,14]. Although we mainly focus on 3G, similar argument can be made for the 4G system as well.

2.1 3G/4G Accounting System Architecture

Figure 1 shows the overall architecture of UMTS/LTE cellular network. The User Equipment (UE, i.e. cellular devices such as smartphones, tablet PCs, etc.) communicates with a target server in the wired Internet by passing the packets through the UMTS, which consists of a Radio Access Network (RAN) and a Core Network (CN). The RAN is responsible for allowing wireless access to the UE and for providing a connection to its CN. Inside the RAN, Node B, a base station for transmitting and receiving data directly with the UE through an air interface, is controlled by a Radio Network Controller (RNC), which manages radio resources and UE mobility. In 4G, the RAN consists of only E-UTRAN Node B (eNodeB) without a RNC since eNodeB also has the control functionality embedded in it.

After passing through the RAN, the packets from a UE enter the General Packet Radio Service (GPRS) through Serving GPRS Support Node (SGSN), which is responsible for delivering packets to or from the UE within its service area. Then, the Gateway GPRS Support Node (GGSN) converts the GPRS packets coming from the SGSN into an appropriate Packet Data Protocol (PDP) format such as IP, and sends them out to an external data network such as the wired Internet where the target server is located. In 4G networks, the basic procedure is the same except for the fact that the SGSN is replaced with a Serving Gateway (S-GW), the GGSN is replaced with a packet data network gateway (P-GW), and the UE's mobility is handled by a mobility management entity (MME).

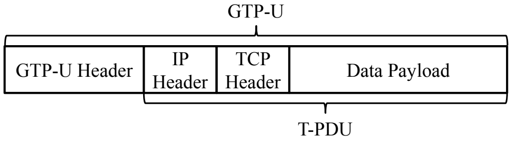


Figure 2: GPRS packet format inside the CN

The cellular data accounting is carried out inside the CN in the form of a Charging Data Record (CDR), which includes the information necessary for billing such as the user identity, the session and the network elements, and services used to support a subscriber session. The CDR is generated by the serving nodes (SGSN, GGSN, S-GW, P-GW) and is forwarded via the Charging Gateway Function (CGF) with the charging information to the Billing System (BS). The CGF can be located anywhere: in an external interface, in every GSN, or in a particular GSN to serve other GSNs.

2.2 3G Accounting Process

When a user establishes a connection with a target server to download some content, it triggers both GSNs to create their own CDRs (S-CDR, G-CDR) related to PDP contexts with the UE's unique charging ID (C-ID) to collect the charging information. The SGSN collects the charging information related with the radio network usage while the GGSN collects that of the external data network usage. The standard charging information collected by GSNs are the radio interface, usage duration, usage of the general packet-switched domain resources, source and destination IP addresses, usage of the external data networks, and the location of the UE.

While the UE downloads its requested content from the target server through the cellular network, the GSNs record the traffic volume arriving to the CN in the form of T-PDU (Figure 2). The T-PDU is the original IP packet received from either the UE or the target server, which is then converted in the CN to move around the GSNs. The T-PDUs are passed between GSN pairs via GTP-U tunnels by attaching the GTP-U header at the front [15]. The accounting process continues until the communication is completed and the UE tears down the connection. When the session is finished, the CDRs stored in the GSNs are forwarded to the BS via the CGF and are processed to calculate the total data volume consumed by the particular session. For byte-level accounting per user, most cellular ISPs account for entire IP packet sizes while their policies differ as to whether they include retransmitted TCP packets or not.

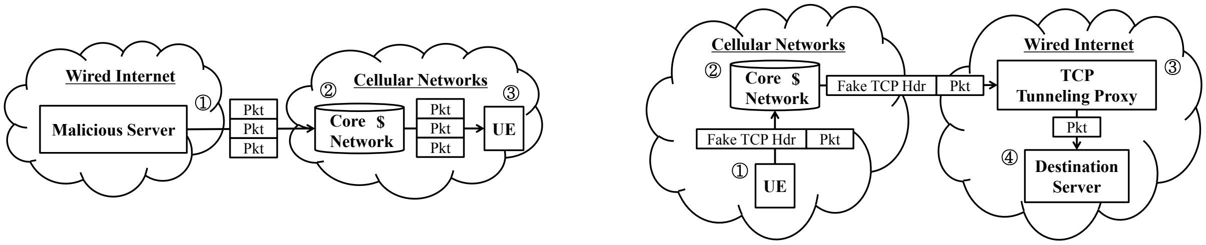
3. ACCOUNTING CHALLENGES IN TCP

In this section, we discuss the accounting issues in TCP-based Internet services in cellular networks. Since the majority of the cellular traffic is based on TCP, accounting of the TCP traffic directly affects the user bill. We first present the service provider's dilemma in accounting for TCP-level retransmission, and discuss the level of retransmission measured in real networks.

3.1 The Cellular Provider's Dilemma

From the cellular ISP's perspective, all headers and payloads from OSI layer 3 and above should be counted as well as retransmissions from layer 4 since they are consuming the cellular network's resources. However, packet retransmissions depend on the network conditions that are typically beyond the control of users. From the perspective of users, the useful data sits in the application layer and only the volume in the application layer should be counted. If the cellular service providers choose the latter, retransmission packets will be treated as a simple overhead. One such implementation is to bypass all retransmission packets whose TCP sequence numbers are older than the next expected sequence number. While this approach is efficient in that it checks only the TCP headers, we find that a naïve implementation is dangerous in practice.

This situation opens up possible attacks on either side of the dilemma as shown in Figure 3. If the provider accounts for the



(a) Usage-inflation attack by a malicious server: 1) A malicious server sends retransmission packets to the client user equipment even if there is no timeout or 3 duplicate ACKs. 2) The core network accounts for all retransmitted packets. 3) The client UE drops all duplicate packets and the application receives only one copy from the OS.

(b) "Free-riding" retransmission attack: 1) The UE attaches a fake TCP header tunneling the real packet and sends it to a TCP proxy. 2) The core network recognizes the packet as retransmission and does not account for it. 3) The TCP tunneling proxy de-tunnels the packet and forwards it to the destination server. 4) The destination server accepts the packet thinking that it is communicating to the TCP proxy.

Figure 3: Attack scenarios that abuse cellular data accounting policies for TCP-level retransmission

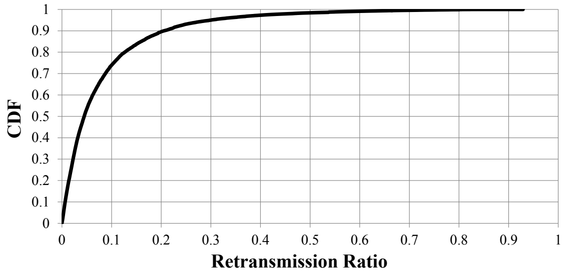


Figure 4: CDF of the retransmission ratios of the flows that experience any packet retransmission in a 3G network

Cellular ISP	Test Client Device
AT&T (US)	Apple iPhone 4 (iOS 5.1.1 - 9B206)
Verizon (US)	Apple iPad 2 (iOS 5.1.1 - 9B206)
SKT (South Korea)	Galaxy S3 (Android 4.0.4)
KT (South Korea)	Galaxy S3 (Android 4.0.4)
LGU+ (South Korea)	Galaxy S3 (Android 4.0.4)

Table 1: Test client devices for each cellular ISP

retransmissions, an attacker can deplete a user's data plan, or if the provider ignores retransmissions, it is possible to tunnel traffic for free if the accounting system does not perform deep packet inspection (DPI). For the latter, we propose to hide our traffic inside of TCP retransmission packets. The basic idea is to send the traffic via a TCP proxy to the destination servers. A mobile client wraps the real TCP traffic in a fake TCP header that looks like a retransmission packet, and sends it to the TCP proxy, and the proxy de-tunnels the real TCP packet and forwards it to the destination. The traffic from the destination is again wrapped in a TCP header that uses an old sequence number by the proxy and is forwarded to the mobile client. This way, a real TCP session can be tunneled in a fake TCP session that avoids accounting.

3.2 Packet Retransmission in Cellular Networks

To determine the level of retransmission in real-world cellular networks, we measured the retransmission ratio at one of the largest cellular ISPs in South Korea. We mirrored the 3G traffic at one of 10 Gbps links just below a GGSN in Seoul, and inspected all TCP flows for 3 hours during the daily peak time (2012/09/29 9PM-0AM). We observed 134,574,018 flows with 6.64 TBs of IPv4 packets. Our monitoring system manages each TCP session by keeping track of TCP connection setup and teardown, sequence numbers, ACKs, and timeouts without a single packet drop during the measurement period.

Overall, we find that the retransmission ratio is reasonably low, which implies that the cellular networks are well-provisioned. Only 1.89% of the flows show a positive number of packet retransmissions during the period. This is in part because the majority of the flows are small (almost 90% of them are smaller than 32 KB) and are short-lived. However, we do find that some large flows experience severe packet retransmissions with as much as 93% of the packets in the flow being retransmitted as shown in Figure 4. This situation can be aggravated by poor provisioning, causing lost or delayed packets at the mobile station and forcing TCP retransmissions.

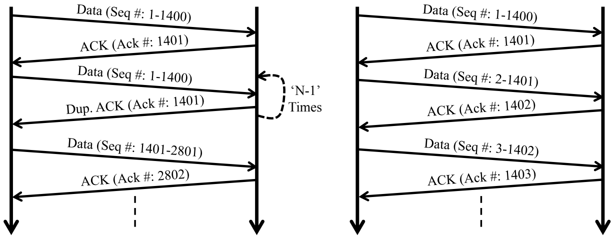
While our measurements imply that accounting for retransmissions would not incur a noticeable usage blowup for most subscribers for now, the users could be the victim of malicious retransmissions that inflate the usage. For example, attackers could participate in popular web sites as advertisers such that their advertisement content is served by a malicious server with a non-compliant TCP stack that intentionally retransmits TCP packets without waiting for timeouts. This way, the attacker can manipulate the accounting mechanism of competing ISPs or blow up the usage of a particular user. In our experiments in Section 4, we show that one can inflate the byte usage arbitrarily if the ISPs blindly account for retransmissions.

4. RETRANSMISSION EXPERIMENTS

In this section, we run various tests to figure out the accounting policies currently being enforced in commercial cellular ISPs. Table 1 shows five large cellular ISPs in the U.S. and South Korea used in our tests as well as the test client devices and their OS versions. We download a file from our custom Web server that manipulates the TCP packets to test a number of retransmission scenarios, and verify whether the accounted volume by the ISP and the byte count in the captured packet trace at clients match.

4.1 Test Setup

To generate retransmission packets at will in the middle of a TCP connection, we build our own server that serves a web object. Our



(a) Controlled retransmissions (b) Quasi retransmissions

Figure 5: Example packet flows of our tests

custom web server accepts a regular TCP connection, processes a Web request, and serves the requested object. When a connection is established, instead of using the accepted TCP socket, the server opens a *raw* socket to read the IP packets from the client by filtering the port and the address, delivers the requested content, and sometimes injects retransmission packets to gauge the accounting policies. This way, we can create our own TCP/IP headers for each outgoing packet and confirm the ACK number from the client. For simplicity, our server maintains a TCP window size of one packet and does not implement congestion nor flow control.

In the client side, we use *wget* to fetch the content from our server. For accurate verification of the accounting volume, we either root or jailbreak our devices and run packet capture programs such as tcpdump [16] or pirni [17]. We collect all packet traces at clients during the test and compare the byte count with the accounted number provided by each ISP. After each download test, we turn off the cellular network interface on the client device and wait until the accounted data volume of the ISP is refreshed. We divide the measured volume into various categories such as normal ACKs without payload, normal data packets, duplicate ACKs, and retransmitted data packets. TCP packets for connection handshake and teardown, and other background traffic are carefully excluded from the results by subtracting them from the total value.

4.2 Experiments and Results

We use five main experiments to determine the accounting policies of various cellular service providers regarding DNS packets and TCP retransmission packets. We include DNS tests to verify the accounting policy loophole reported by recent works [9,10] and to reflect the policy into the measured results. Each test is run three times and we show the average value. The ISPs are addressed by number, with ISP-1, 2 and 3 based in South Korea and ISP-4 and ISP-5 being based in the United States. We note that ISP-1 and ISP-4 provide an accounting granularity of 1 KB, ISP-2 and ISP-3, a granularity of 100 KB, and ISP-5, a granularity of 1 MB.

4.2.1 DNS Packet Accounting

Peng et. al. recently report that packets with the DNS port are considered as a free service and are not accounted for in a number of ISPs [9, 10]. Our first step is to verify this claim by running DNS lookups of 10,000 different domain names and comparing the data volume seen by the client and by the ISP. In our measurements in October 2012, we found that ISPs 1, 2 and 3 do not account for UDP-based DNS packets, but we were surprised to discover that ISP-4 and 5 account for all DNS packets, suggesting that some providers have already started to react to the DNS tunneling reports. In addition, we check whether the TCP packets going through port 53 (DNS) are considered free by downloading some content on the DNS port. We confirm that ISPs 1, 2, 3 that do not account for UDP-based DNS packets do charge for all TCP packets on port 53, thus DNS tunneling attacks are not possible with these ISPs.

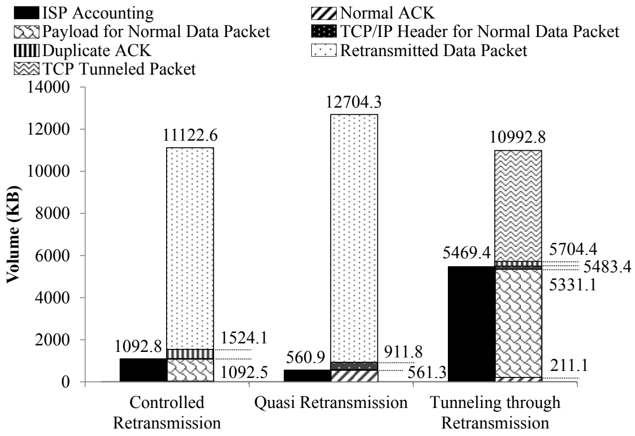


Figure 6: Experiment results of ISP-1

4.2.2 Content Transfer without Packet Loss

As a base case, we compare the measurement results by downloading a file over a reliable link without any intentional packet retransmissions. This test is to verify whether the ISPs account for the traffic accurately in a normal situation with little packet loss. We calculate the theoretical value and compare it with the ISP's accounted volume as explained below. For each test, we confirm the absence of packet retransmissions by checking the captured packet traces. We compare the accounting values from three sources; the ISP, the mobile client, and the theoretical model. We calculate the theoretical value by taking into account the TCP connection setup/teardown, ACK and data packets including headers from layer 3 and up, and the background traffic from other local processes running on the mobile client. We find that all ISPs account for the proper amount of the data volume in this test, confirming the accurate accounting in the base case.

4.2.3 Controlled Retransmissions

We also run the test that intentionally injects retransmission packets between each data packet. We initiate a TCP connection from the mobile client, make sure that the server goes through the TCP-handshake, and then have the server send a pre-determined number of retransmission packets per each data packet. From the size of the original data to be transmitted, we can easily calculate the total volume, as the retransmissions will act as a simple multiplier. Figure 5(a) shows the test scenario.

We wait for the mobile device to ACK a retransmitted packet before sending another one to ensure proper reception. We test each ISP with 9 retransmissions per each data packet (e.g., 10 identical data packets in total, a blowup by a factor of 10 in the real payload).

We discover that only two ISPs in South Korea (ISP-1 and ISP-2) do not account for the retransmission packets while the others do. The two leftmost bars in Figure 6 and Figure 7 show the results of ISP-1 and ISP-2. We download a 1 MB file for ISP-1 while we use a 10 MB file for ISP-2 since ISP-2 supports 100 KB accounting granularity. Interestingly, we see that the accounting policy differs from ISP to ISP even in the same country. ISP-3 in South Korea accounts for every packet regardless of retransmission. We also note that the accounting policies for ISP-1 and ISP-2 are slightly different. While they both ignore retransmitted data packets, ISP-2 accounts for duplicate ACKs while ISP-1 ignore them for accounting. We confirm that the other three ISPs count every retransmission, showing a blowup by a factor of 10 from the original file size in the accounted volume. For this reason, we leave out the graphs for these ISPs. This test implies that the users in these ISPs can be the victim of usage-inflation attack.

4.2.4 *Quasi Retransmissions*

The next question is how the service providers would account for partial retransmissions where the next packet overlaps partially with the previous packet. More specifically, the server increments the current window by just one byte, but the data content is much larger. The resulting stream is one where the sequence numbers are not directly repeated, but the data content largely overlaps. This will tell us if the service provider accounts by data packets, or takes the actual data window of a TCP packet into account. We send a small amount of application layer data (10KB, 75KB), but make sure that the packet window is only incremented by one byte, although the payload of each packet contains over 1.3 KB of content. We omit the ISPs that charge for retransmissions since they only account for the complete volume anyway.

The two middle bars in Figure 6 show the result for ISP-1. We see that the accounted value is actually less than the data volume excluding the retransmitted data packets. This is due to the ISP not charging the TCP/IP headers for data with partially-retransmitted payload. ACK packets are all counted since each ACK packet has its acknowledgement number increased by one. On the contrary, ISP-2 (middle bars in Figure 7) accounts for all TCP/IP headers but not the retransmitted payload itself. This could be explained by an ISP that checks the sequence number and the packet length to identify the actual data volume but charges for the entire header since there is at least one byte of new payload.

4.2.5 *Tunneling through Retransmissions*

Finally, we measure if the service providers verify that the data content of retransmissions do in fact contain a copy of the previous packet's payload data. If they only rely on the TCP headers, an attacker could set up a covert channel in the payload field of the TCP retransmission packets to avoid data charges. We were also careful to set the sequence number of the retransmission packets to be within the range of the most recently-ACK'ed packet to prevent middleboxes or the recipient's OS kernel from dropping packets with old sequence numbers.

The two rightmost bars in Figures 6 and 7 show that both ISP-1 and ISP-2 do not account for retransmitted packets with different payload. This makes intuitive sense since deep inspection of the TCP payload of every packet would be space and time consuming. From this test, we conclude that all ISPs that do not account for retransmitted packets are open to TCP-retransmission tunneling.

5. MITIGATION

To provide fair accounting, one can decide to account for retransmitted packets but block the "usage-inflation" attack or decide not to account for retransmitted packets but defend against the "free-riding" attack. The former makes sense if we can assume a low legitimate packet loss rate throughout the cellular infrastructure, but it could penalize users that are already getting poor coverage service. Instead, we focus on the latter here and briefly propose three plausible mitigation techniques against "free-riders".

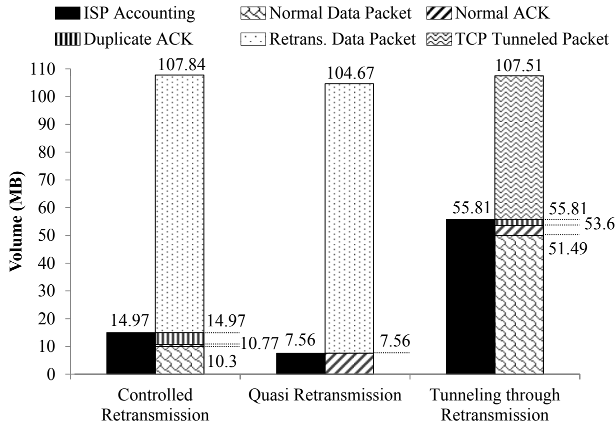


Figure 7: Experiment results of ISP-2

Detection of Abnormal Retransmission. The cellular ISP may set a limit on the number or ratio of retransmission packets per flow. The GSN detects an abnormal flow with the number of retransmissions exceeding a certain threshold, and alerts the ISP of a possible attack. Once a flow turns out to abuse the retransmission policy, the ISP can decide to either charge all retransmission packets or explicitly close the connection. This approach is attractive since it requires only small states per each flow (e.g., number of retransmissions per packet, retransmission ratio, etc.), not causing much overhead on the CN. However, the major disadvantage of this method is that it could incur a false-positive alarm. We have shown that even a legitimate flow experiences the retransmission ratio of 93% in poor cellular network environments. Therefore, naively setting a threshold could result in penalizing innocent users or tunneling attacks, depending on the value of the threshold.

Deterministic DPI. A more accurate solution is to run DPI on all TCP flows where the system temporarily stores the content in every flow and compares the payload if it detects a retransmission. A bytewise comparison over the retransmitted range can ensure identical retransmission content. This method is advantageous in that it can completely remove the false-positive alarm. One obvious drawback, however, is that it could incur high system overheads of managing the buffer of every TCP flow. In our measurement at a 10 Gbps 3G backbone link, we see up to 1.3 million new TCP flows per minute with 270 K concurrent flows at peak. In the worst case, an accounting system should handle tens of millions of packets per second per 10 Gbps link. We are currently building a middlebox system that can manage 100Ks of concurrent flows for a 10 Gbps link by careful buffer memory management and parallel processing on a multicore system. However, it would be still challenging or costly if it requires multiple load-balanced machines.

As a hybrid solution, one might set a small threshold for detecting abnormal retransmissions and run deterministic DPI only if the retransmission ratio is beyond the threshold. This would greatly reduce the system overhead by bypassing the majority of normal flows, and could detect long-lived flows that tunnel packets. However, it is still not perfect if a sophisticated attacker carefully manages small flows that do not trigger the alarm.

Probabilistic DPI. To reduce the memory requirements, we propose a lightweight method where the CN only has to inspect a random part of the TCP payload. Thus, between two candidate retransmissions, this method needs to (1) look in the same places in the payload and (2) find the same bytes at those positions. For step (1), we can use the sequence number as an index into a table containing *n* random locations per packet where bytes will be extracted from the payload. We could use a random number generator with a secret number as a seed to determine the *n*-byte locations per each flow. For step (2), once we have extracted some bytes from the same location on both packets, we can compute the difference between those *n*-byte sequences. If it is anything other than 0, we can confirm that the retransmission payload is different from the original payload.

We note that we have only reduced the space complexity by a constant factor, from a full TCP payload to an *n*-byte representation, but storing a fraction of the payloads at minimal computing costs will help in the real-world implementation. The probability of collisions between the original payload and an arbitrary payload decreases exponentially as *n* increases, making the scheme quite space efficient. We also note that it would increase false negatives if *n* is too small. We plan to identify the appropriate *n* by analyzing multiple variables in TCP flows, including the average payload length, the probability of overlapping sequence number ranges in the retransmitted packets, the number of concurrent flows, and the average congestion window size.

6. RELATED WORKS

Peng et. al. have recently reported loopholes in some cellular ISPs that allow attackers to obtain free cellular Internet access by tunneling the data on the DNS port, since DNS is viewed as a free infrastructure service and the payloads are not inspected [9, 10]. In our measurements, we find that TCP packets on the DNS port get charged even for the ISPs that allow free UDP-based DNS packets. Running DPI on the DNS packet would incur relatively small overheads since the number of DNS packets is typically much smaller that that of other data packets and each DNS packet is just a few hundred bytes. In addition, we have shown that there is a more fundamental issue in cellular data accounting for TCP-level packet retransmission. Building a DPI-based cellular accounting system that analyzes every TCP packet going through the CN remains to be a challenge.

Lee provides one of the early works that measure the retransmission rate over CDMA 1x EV-DO service [18]. The author finds that the average retransmission rate of a flow reaches up to 4.7% with 92% burst retransmissions in the uplink. The retransmission ratio shows a similar characteristic to our experiment where larger flows are more likely to be affected by the retransmissions. Won et. al. show that in CDMA networks, almost 80% of the total packets captured in the link are retransmission packets [19]. They also find that 38% of the TCP sessions have 9 out of 10 packets as retransmission packets, which implies that our attack could be effective in the CDMA network as well. Jang et. al. look at the retransmissions in HSDPA networks (3G, 3.5G) from moving cars and express trains [20]. Their results show that when the vehicles are on the move, the retransmission ratio increases up to 71 times higher than that in the stationary case, implying that the users with higher mobility will have to pay more if retransmission packets are accounted. Gember et. al. measure the retransmission rate in Wi-Fi networks at a university campus [21]. They show that even in the less-congested wireless network, 5% of flows have one or more retransmission packets where 80% of them are due to packet loss.

7. CONCLUSION

We have shown that due to the current design of the cellular data architecture and transport layer reliability mechanisms using retransmissions, the accounting policies either leave the user vulnerable to data depletion attacks, or cause the ISP to be vulnerable to service charge evasion due to tunneling through retransmissions. We have measured the effect of retransmissions on five major ISPs in two countries, demonstrating the possibility of data depletion attacks, or free-riding tunneling. We believe that it is possible for ISPs to provide a fair accounting of traffic usage while preventing free-of-charge abuse, and have proposed possible mitigations that could be implemented with relatively low costs.

8. ACKNOWLEDGMENTS

We appreciate insightful comments from anonymous reviewers and thank our shepherd, Ramon Caceres, for his help on improving the paper. This research was supported by the KCC (Korea Communications Commission), Korea, under the R&D program supervised by the KCA (Korea Communications Agency), KCA-2013-12911-05003.

9. REFERENCES

[1] CISCO. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016. Technical report, 2012.

[2] ITU. ICT Facts and Figures. Technical report, 2011.

[3] Ericsson. Traffic and Market Report. Technical report, 2012.

[4] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perälä. HSDPA Performance in Live Networks. In *Proceedings of IEEE International Conference on Communications*, 2007.

[5] G. Maier, F. Schneider, and A. Feldmann. A First Look at Mobile Hand-Held Device Traffic. In *Proceedings of the Passive and Active Measurement*, 2010.

[6] F. Qian, K. S. Quah, J. Huang, J. Erman, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Web caching on smartphones: Ideal vs. reality. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, 2012.

[7] J. Erman, A. Gerber, K. K. Ramakrishnan, S. Sen, and O. Spatscheck. Over The Top Video: The Gorilla in Cellular Networks. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement Conference*, 2011.

[8] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[9] C. Peng, G. Tu, C. Li, and S. Lu. Can We Pay for What We Get in 3G Data Access? In *Proceedings of Annual International Conference on Mobile Computing and Networking*, 2012.

[10] C. Peng, C. Li, G. Tu, S. Lu, and L. Zhang. Mobile Data Charging: New Attacks and Countermeasures. In *Proceedings of ACM Conference on Computer and Communications Security*, 2012.

[11] 3GPP. UMTS. http://www.3gpp.org/Technologies/Keywords-Acronyms/article/umts.

[12] 3GPP. LTE. http://www.3gpp.org/LTE/.

[13] 3GPP. ETSI TS 132 200. Telecommmunication management; Charging management; Charging principles.

[14] 3GPP. ETSI TS 132 215. Telecommunication management; Charging management; Charging data description for the PS domain.

[15] 3GPP. ETSI TS 129 060. General Packet Radio Service; GPRS Tunnelling Protocol across the Gn and Gp interface.

[16] Gadgetcat. tcpdump on Android, 2011. http://gadgetcat.wordpress.com/2011/09/11/tcpdump-on-android/.

[17] n1mda-dev. Pirni native iPhone ARP spoofer and network sniffer. http://code.google.com/p/n1mda-dev/wiki/PirniUsageGuide.

[18] Y. Lee. Measured TCP Performance in CDMA 1x EV-DO Network. In *Proceedings of the Passive and Active Measurement*, 2005.

[19] Y. J. Won, B. Park, S. Hong, K. Jung, H. Ju, and J. Hong. Measurement Analysis of Mobile Data Networks. In *Proceedings of the Passive and Active Measurement*, 2007.

[20] K. Jang, M. Han, S. Cho, H. Ryu, J. Lee, Y. Lee, and S. Moon. 3G and 3.5G Wireless Network Performance Measured from Moving Cars and High-Speed Trains. In *Proceedings of ACM Workshop on Mobile Internet Through Cellular Networks: Operations, Challenges and Solutions*, 2009.

[21] A. Gember, A. Anand, and A. Akella. A Comparative Study of Handheld and Non-Handheld Traffic in Campus Wi-Fi Networks. In *Proceedings of the Passive and Active Measurement*, 2011.