Article

Improving TCP Performance in Vehicle-To-Grid (V2G) Communication

Jinwoo Park 1, Hyogon Kim 2 and Jin-Young Choi 1,*

1 Graduate School of Information Security, Korea University, Seoul 02841, Korea; jwpark@formal.korea.ac.kr
2 Department of Computer Science, Korea University, Seoul 02841, Korea; hyogon@korea.ac.kr
* Correspondence: choi@formal.korea.ac.kr; Tel.: +82-2-3290-3575

Received: 30 September 2019; Accepted: 17 October 2019; Published: 23 October 2019

Abstract: On a connected car, the performance of Internet access will significantly affect the user experience. For electric cars that use vehicle-to-grid (V2G) communication to interact with the Internet during charging, the charge cable quality poses a challenge to the V2G communication. Specifically, the performance of Transmission Control Protocol (TCP), the transport protocol that most Internet applications use, may suffer due to the high noise and consequent errors that the charge cable presents. Currently, TCP NewReno is the TCP implementation that ISO 15118 standard stipulates for the V2G communication. However, its congestion control algorithm has been designed for the general Internet environment where congestion, not link errors, account for most of packet losses. Indeed, we confirm that the throughput of TCP NewReno rapidly degrades as the error rate increases on the charge cable. Specifically, we show that other TCP variants such as TCP Illinois far exceeds TCP NewReno in both lossy and non-lossy link environments. Finally, we propose how to configure TCP NewReno parameters to make it achieve the throughput comparable to other TCP variants on V2G communication environments, regardless of the link quality presented by the charging cable.

Keywords: V2G; TCP congestion control; connected car

1. Introduction

In recent years, automobiles are evolving into information devices, and they are increasingly connected to the Internet. Some expect that, by 2020, connected cars will account for 75% of the world’s new cars [1]. Connected cars can provide infotainment services that require more complex data such as gas station pricing, music viewing, video viewing, and games. To facilitate the user experience on a connected car, the performance of the Internet should be high. Today, most Internet applications use Transmission Control Protocol (TCP) as the transport. Therefore, the performance of TCP plays a crucial role in the connected car environments.

Currently, TCP is being introduced in various standards of automobile communication: in-vehicle, wireless vehicle-to-everything (V2X), and vehicle-to-grid (V2G) [2–4], among others. Probably for the consistency among vehicle-related communication standards, they require the same TCP implementation, TCP NewReno. For one, the ISO 15118 standard for the V2G communication stipulates that TCP NewReno should be used in these standards. Specifically, the ISO 15118-2 specification refers to the one defined in RFC 6582. Likewise, AUTOSAR, a standard software architecture for the communication between electronic control units (ECUs), has introduced TCP in its standard specifications. AUTOSAR Specification of TCP/IP Stack [2] specifically requires NewReno as defined in RFC 6582 [5].

TCP NewReno is a congestion control algorithm developed in 1999 [6]. Considering the conservative nature of the automobile industry that puts the safety at the highest priority, it is understandable that they chose the time-tested technology. However, TCP NewReno has developed for
the traditional Internet environment where congestion, not link errors, account for most of packet losses. For the V2G environment, with the power-line communication link that has the latter characteristics, it may not fit well. Moreover, the experience with various environments such as wireless Internet has led to more advanced implementations than NewReno. Therefore, we need to investigate if the decision to employ TCP NewReno in the V2G communication is warranted, and, if not, what can be done to improve the TCP performance in the V2G environment.

In the reset of this paper, we show that the throughput of TCP NewReno rapidly degrades as the error rate increases on the charge cable. We find that more recent TCP implementations such as TCP Illinois far exceed TCP NewReno in both lossy and non-lossy link environments. Specifically, TCP Illinois achieves more than twice the throughput of NewReno in the worst link condition. However, even the best performing TCP implementation leaves room for further improvement because their designed target environment does not fit well with V2G. We propose a modification to TCP Illinois that leads to further throughput increase. Finally, we also investigate how to configure TCP NewReno parameters to make it achieve the throughput comparable to other TCP variants on V2G communication environments, regardless of the link quality presented by the charging cable.

2. Related Work

2.1. Background

2.1.1. V2G Communication Standards

V2G communication is expected to be available for automatic billing and Internet access during charging when electric cars use public charging stations. In this expectation, the international standard ISO 15118 has been established to facilitate efficient V2G communication and global compatibility. The major components that make up the V2G system as envisioned in ISO 15118 are as shown in Figure 1. The Supply Equipment Communication Controller (SECC) is the communication controller on the charging service provider side. The Secondary Actor (SA) is a server on the Internet that provides services such as billing and other Internet services. The Electric Vehicle Communication Controller (EVCC) is the communication controller on the electric vehicle side. Connecting the EVCC and SECC is the Power Line Communication (PLC) link that carries both the charging current and the communication signals.

![Figure 1. V2G communication environment as assumed by ISO 15118.](image)

The ISO 15118 standard suite is subdivided into ISO 15118-1, 15118-2 and 15118-3 [4,7,8], as illustrated in Figure 2. ISO 15118-1 describes general requirements and use case definitions for V2G communications. It provides service certification, target confirmation, charging start and stop procedures for charging an electric car. ISO 15118-2 defines the message types and formats between EVCC and SECC to support the use cases defined in 15118-1 for the network and application layer requirements for V2G communications. In addition, IPv6, TCP, TLS, EXI, and XML are defined as protocol components. In particular, the TCP implementations discussed in this paper relate to the ISO 15118-2 standard. ISO 15118-3 is requirements on the physical layer and data link layer of V2G communication. It defines several requirements for charging an electric car based on the PLC.
2.1.2. Power Line Communication (PLC)

PLC is a communication technology that uses a power line as the communication medium. The biggest advantage of PLC is that it can communicate with the power line in use, so there is no additional wiring work, which reduces cost. In smart charging, the authentication between the electric car and the charging service provider is automatically performed by using the physically connected power line.

The PLC link used for SECC and EVCC uses ISO 15118-3: G33-PLC based on ITU-T G9903 [9] and HomePlug Green PHY (GP) [10]. The G3-PLC uses a narrow frequency band of 3–500 kHz, with data rates up to several hundred kbps. HomePlug GP uses a wide frequency band of 1.8–30 MHz, with data rates up to 10 Mbps. Many automobile companies in the US and Europe are adopting HomePlug GP [11]. In PLC, the charging cable is not a medium originally manufactured for communication. Moreover, there is much noise. The main ones are background noise and impulse noise that occur when the load starts and stops or when the electric vehicle starts or turns off. The impulse noise, in particular, is a major source of packet errors in PLC [12,13].

2.2. Comparative Works on TCP Implementations for Lossy Links

TCP congestion control has been studied for more than thirty years, and it is impossible to list all of the existing works in this paper. Table 1 lists some well-known implementations with their target operating environment and key features.

The question of which to use among the myriad of TCP implementations largely depends on the particular communication environment where the vehicular communication takes place. For example, V2X uses a wireless link, AUTOSAR deals with links with very small delay time and packet loss rate, and V2G utilizes PLC where random and bursty errors occurs due to sudden interference [14]. In this paper, we consider whether the decision to choose TCP NewReno over others for V2G communication is desirable in terms of performance. Furthermore, we explore what we can do to resolve possible performance issues caused by the decision.
Table 1. Characteristics of some TCP implementations.

| Implementation | Target  | Classification | Salient Features                                                                 |
|---------------|---------|----------------|----------------------------------------------------------------------------------|
| NewReno [15]  | low BDP | loss-based     | Fast Recovery; avoids timeout on partial ACKs and makes progress when multiple packet losses occur in a window |
| Vegas [16]    | low BDP | delay-based    | detects congestion by end-to-end delay                                             |
| Veno [17]     | wireless| delay-based    | improves on Vegas; tries to distinguish congestion-induced losses from link-induced losses |
| H-TCP [18]    | high BDP| loss-based     | seeks to increase the aggressiveness of TCP on high bandwidth-delay product (BDP) paths |
| Highspeed [19]| high BDP| loss-based     | grows window faster than standard TCP under large congestion window size           |
| Hybla [20]    | satellite| delay-based    | aims to eliminate penalization of TCP connections with very long RTTs             |
| BIC [21]      | high BDP| loss-based     | uses a binary search algorithm to find the largest window size that will last the longest |
| Scalable [22] | high BDP| loss-based     | recovers window much more quickly from loss; window reduction by a smaller fraction, increase by a slower rate |
| Cubic [23]    | high BDP| loss-based     | improves on BIC, less aggressive; window is a cubic function of time since last congestion |
| Illinois [24] | high BDP| hybrid         | window increase and decrease are both decreasing functions of queuing delay        |
| Westwood [25] | wireless| delay-based    | distinguishes between congestive and non-congestive losses                        |
| YeAH [26]     | high BDP| hybrid         | balances between efficiency, fairness, friendliness to Reno, induced network stress, robustness to random losses |

Thus far, there have been a couple of comparison studies between different TCP implementations for the connections that involve lossy links, none of which consider PLC links. Caini et al. [27] compared the performance of TCP implementations in satellite communication environments. In their experiments, the satellite link was set to have a long round-trip time (RTT) of 25–600 ms, and the average packet loss rate was set to 0–1%. The authors found that, regardless of the average packet loss rate, TCP Hybla always had the best performance when RTT was very large. They also found that TCP Illinois achieves a final goodput very close to the best case in the given environment.

Tsinknas et al. [28] compared the performance of TCP congestion control algorithms in WiMAX and wireless LAN (WLAN) environments. It is assumed that the packet loss occurs in the wireless link without any packet loss in the wired section. The goodput was measured while varying the average packet loss rate from 0.001% to 5%. The channel bandwidth was set to 7 MHz and the frame length to 5 ms. Among NewReno, Vegas, Veno, Westwood, and BIC, Westwood showed the best performance, with 96% more goodput than NewReno. Regardless of the packet loss rate, the BIC algorithm also showed about 18% higher performance than NewReno on average.

As we can observe in Table 1, the loss, delay, and bandwidth conditions given by the communication environment all affect the choice of the fittest TCP implementations. Although the TCP connections over satellite or other wireless links share a common aspect of lossy links, the bandwidth or delay characteristics are different. For instance, the RTTs on V2G are expected to be not as high as those of satellite links, and the total bandwidth on a PLC link is limited to 10 Mbps [10], much smaller than in typical WLANs today. Therefore, we need to newly conduct a study that more closely simulates the PLC link over which TCP connections are carried.
3. Performance Comparison of TCP Implementations for V2G Communication

In the V2G communication scenario, as depicted in Figure 1, the SA is most likely not an embedded machine but a server that runs on a general purpose operating system (OS) such as Linux. These general purpose OSs can implement many TCP variants as Linux does [29], one of which it uses as a default. For ease of experiments, however, we instead resort to the NetworkSimulator3 (NS-3) simulator that also implements as many as ten TCP variants: NewReno, Hybla, Highspeed, Htcp, Vegas, Veno, Scalable, Illinois, Bic, and Westwood.

For the V2G connection topology, we simplify it as in Figure 3. In the experimental environment, the server SA on the Internet transmits data. The transmitted data are received by EVCC, which is the communication controller of the electric vehicle, through the SECC, which is the communication controller of the electric supplier and is the charger. We assume that the PLC link between SECC and EVCC with 10 Mbps of bandwidth and 0.9 ms of latency is the bottleneck, which is the maximum transmission rate of HomePlug GP. In contrast, we assume that the section between SA and SECC has a larger bandwidth, which is 500 Mbps and 0.9 ms of latency. Therefore, any congestion loss would also takes place at the output buffer of the SECC. We also assume that the non-PLC section has a larger delay than the PLC link. Although Figure 3 shows the topology for a single TCP connection between the EVCC and SA, we use the same configuration in terms of bandwidth and delay also for more complex topology that accommodates a larger number of electric vehicles in the charging facility. In particular, we vary the number of electric vehicles connected to the charging station and connected to the Internet from 1 to 21.

As to the data flow, we assume that the SA sends most of traffic to EVCC, much more than in the opposite direction. The other set values of NS3 are as follows. The size of the TCP send and receive buffers are set to 128 kbytes, which is the default setting of NS3. The maximum transmission unit (MTU) is set to 1500 bytes, thus the TCP maximum segment size (MSS) is set to 1460 bytes.

Finally, we vary the non-congestive loss rate on the PLC link from 0.25% to 5%. Note that this is an average rate, because the packet losses on the PLC link can occur suddenly and randomly due to, e.g., the impulse noise [30]. To model this, we used BurstErrorModel, a bursty error model of NS3. In this model, packets are errored corresponding to an underlying distribution, burst rate, and burst size. For each received packet, the PLC link determines if it is a start of a new loss burst. In our experiments below, we set it at 0–5%. Once the burst starts, we let the burst size randomly determined between 1 and 4, making the average burst length 2.5 packets.

3.1. A Single V2G Connection

In a charging station, there can be few or many vehicles connected for charging. We look into these both cases. We first investigate one extreme case of a single TCP connection between the electric vehicle and the SA. Namely, there is one electric car connected to the charging circuit. In this situation, we measure the throughput performance of each TCP implementation. Table 2 shows the throughput for each TCP implementation when SA transmits varying data size from 10 Mbytes to 800 Mbytes, when the packet error rate is 5% on the PLC link. Throughput is the average of six simulation runs. We first notice that there is no major difference in throughput as we vary the transfer size.
Table 2. Average throughput (Mbps) of TCP implementations for different transfer size, packet loss rate = 5%.

| File Size | NewReno | Hybla | Highspeed | Htcp | Vegas | Scalable | Veno | BIC | Illinois | Westwood |
|-----------|---------|-------|-----------|------|-------|----------|------|-----|----------|----------|
| 10 MB     | 0.418   | 0.851 | 0.925     | 0.426| 0.755 | 1.316    | 0.506| 0.697| 2.294    | 0.385    |
| 50 MB     | 0.453   | 0.890 | 0.976     | 0.482| 0.821 | 1.333    | 0.534| 0.738| 2.332    | 0.435    |
| 100 MB    | 0.485   | 0.947 | 0.968     | 0.506| 0.765 | 1.396    | 0.544| 0.755| 2.266    | 0.471    |
| 200 MB    | 0.454   | 0.933 | 0.968     | 0.511| 0.790 | 1.371    | 0.543| 0.723| 2.334    | 0.421    |
| 400 MB    | 0.455   | 0.934 | 0.926     | 0.505| 0.767 | 1.368    | 0.542| 0.705| 2.228    | 0.428    |
| 600 MB    | 0.455   | 0.943 | 0.957     | 0.506| 0.772 | 1.395    | 0.546| 0.721| 2.297    | 0.427    |
| 800 MB    | 0.455   | 0.954 | 0.954     | 0.502| 0.768 | 1.385    | 0.546| 0.719| 2.294    | 0.426    |

Second, we notice that Illinois achieves far better throughput than any other TCP implementation. This is a noticeable result because, under far higher RTT values, TCP Hybla performs best [27]. It shows that the bottleneck bandwidth and the RTT indeed affect the performance of TCP implementations, which we need to heed when we select the one for the V2G communication.

As we find that the transfer size does not significantly affect the throughput in all tested TCP implementations, we pick one size, 100 Mbytes, for closer analysis. With this transfer size, Figure 4 shows the throughput measured while varying the average packet loss rate from 0.25% to 5%. We can observe that, due to bursty losses, the throughput quickly degrades in most TCP implementations even when the average packet loss rate itself is capped at 5%. However, among the ten implementations, TCP Illinois had the best throughput, under virtually all average packet loss rates. In contrast, TCP NewReno is among the worst performing implementations. It is above only one or two other implementations.

Compared with TCP Illinois, it fares relatively well when the packet loss rate is very low. When the loss rate is 0.25%, the throughput is 7.526 Mbps for NewReno and 9.323 Mbps for Illinois. Even under this favorable condition, TCP Illinois proved to have about 23% better throughput than TCP NewReno. However, the performance gap increasingly widens as the loss rate on the PLC link worsens. When the average packet loss rate is as high as 5%, the throughput is 0.485 Mbps for NewReno and 2.266 Mbps for Illinois. Namely, Illinois attains approximately 3.7 times higher throughput than NewReno.

Figure 4. Throughputs as functions of average packet loss rate for a single TCP connection; transfer size = 100 MB.
3.2. Multiple V2G Connections

Now, we consider the second case where several electric cars are charging at the same time. According to Tesla, they have more than 13,344 electric car chargers installed in 1533 charging stations. That is almost nine chargers per charging station [31]. However, if the supply of electric cars increases, more chargers will be installed per charging station. Therefore, this paper assumes a situation where the number of chargers per charging station is as many as 21. Namely, there are 21 EVCCs. To find the throughput performance of TCP implementations in such a situation, we construct the simulation configuration as shown in Figure 5. Here, SECCs share LAN segments to connect to remote SAs for their respective EVCCs. In this simulation, three SECCs share the 10-Mbps bandwidth on the PLC link.

![Simulation setup for 21 electric vehicles.](image)

Again, we assume that SA sends 100 Mbytes of data to its corresponding EVCC. Figure 6 shows the average throughput per vehicle as functions of packet loss rate. We first notice that the maximum throughput is scaled down by the number of EVCCs per 10 Mbps links. Secondly, we notice that TCP Illinois again has the best throughput performance. TCP Scalable produces the second highest throughput. Comparing NewReno with Illinois, they have 2.702 Mbps and 2.920 Mbps, respectively, for the average packet loss rate of 0.25%. TCP Illinois has approximately 8% higher throughput performance than NewReno. However, as the loss rate increases, the performance gap is widened. The throughput for Illinois is 1.454 Mbps, outshadowing 0.422 Mbps of NewReno for 5% loss ratio. The throughput is 2.4 times higher by using TCP Illinois instead of NewReno.

Figure 7 shows the sequence number progressions during the first minute of the transfer, when the average packet loss rate is 5%. We first observe that TCP Illinois proceeds the fastest. It means that other TCP implementations are not aggressive enough to exploit the network capacity in the face of bursty errors. Second, the sequence number progression for NewReno reveals that it is the second most conservative variant among the ten compared algorithms. Such conservativeness does not appear to fit well with the inevitable PLC link that inflicts bursty losses in the V2G communication.
3.3. On Aggressiveness of NewReno and Illinois

The aggressiveness/conservativeness of TCP NewReno and Illinois are determined by two mechanisms used on the packet loss event. In the case the retransmission timer expires, both algorithms execute Slow Start. On the other hand, if three duplicate ACKs are received, NewReno reduces the congestion window size by half. However, Illinois reduces the congestion window size by a factor between 0.125 and 0.5, according to the arrival time of each packet. Namely, it is more aggressive than NewReno in the case there are surviving packets in the window after the lost ones, which means mild congestion.

Although more aggressive than NewReno, the Illinois algorithm additionally uses the delay as a second measure of how congested the TCP connection path is. It enables the algorithm to exert...
more cautious control, while being aggressive. Figure 8 compares the number of Slow Start and Fast Recovery events in these two TCP implementations. Note that the events are triggered by the loss detection by the retransmission timeout and the Fast Retransmit, respectively. We can observe in the figure that the total number of these events is much smaller with TCP Illinois, telling us that TCP Illinois is using more cautious control. Moreover, the ratio of Fast Recovery to Slow Start events is much larger in TCP Illinois, which means that more packet losses are not as severe as in TCP NewReno.

![Graph showing comparison between TCP Illinois and TCP NewReno]

*Figure 8. Slow Start and Fast Recovery events in NewReno and Illinois, for 100 MB transfer.*

4. Improving TCP Performance for V2G Communication

The lessons from the previous section is that we need more aggressiveness in the TCP congestion control algorithm in the face of the bursty losses inflicted by the PLC link. Moreover, we also observe that the increased aggressiveness does not adversely affect the throughput performance since the performance with TCP Illinois at the lowest packet loss rate is still higher than the NewReno implementation. In this section, we explore if and how we can improve the TCP performance for V2G communication by imparting more aggressiveness in the TCP implementations. For the exploration, we take two different approaches:
1. Take the best performing TCP implementation, i.e., Illinois, and increase its aggressiveness.
2. Take the standard TCP implementation for V2G, i.e., NewReno, and increase its aggressiveness.

The reason that we further increase the aggressiveness of already aggressive Illinois is because Illinois is not designed with the V2G communication path as the target operating environment. Therefore, we investigate if there is room for more aggressiveness, hence higher throughput, for the TCP variant in the given V2G communication environment. The justification for the second approach, on the other hand, is based on the fact that TCP NewReno has already been selected as the standard implementation for V2G, not to mention other vehicular communication standards such as AUTOSAR. Given that the standard stays, we need to explore if we can improve the TCP performance in the V2G context by fine tuning the congestion control parameters in the implementation.

4.1. Modifying TCP Illinois for V2G Communication

Let us start by looking into TCP Illinois. We proceed by first understanding the Illinois algorithm, and then modifying it.

4.1.1. Original TCP Illinois

Similar to TCP NewReno, TCP Illinois is also based on the AIMD (Additive Increase Multiplicative Decrease) control method [32,33]. When the network is not congested, the AI component additively increases the transmission rate by a predetermined factor $\alpha$ for each received ACK. However, when it is determined that the network is congested, the MD component multiplies the transmission rate by a predetermined factor $\beta$. Namely, the AIMD method is described by the following equations:

\[
W \leftarrow W + \frac{\alpha}{W}, \text{ on every ACK} \tag{1}
\]

\[
W \leftarrow W - \beta W, \text{ on 3 duplicate ACKs} \tag{2}
\]

\[
W \leftarrow 1, \text{ on Slow Start} \tag{3}
\]

Notice that, in NewReno, $\alpha = 1$ and $\beta = 0.5$. In other words, these factors remain static. TCP Illinois, however, changes it so that they are functions of the average queueing delay on the TCP path. The modified AIMD is called the Loss-Delay based Congestion Avoidance, Concave-AIMD (LDCA-CAIMCD). The differences between TCP NewReno and Illinois in terms of the $\alpha$ and $\beta$ values are illustrated in Figure 9.

Notice that Illinois regulates the two parameters based on the measured queueing delay—they are not static as in NewReno. The delay is a very important congestion signal, second only to packet losses. We can interpret the increased queueing delay as a signal of an imminent congestion. Specifically, the average delay is compared with the delay of the newly sent packet. If the delay of the newly sent packet is shorter, it is determined that the packet is not congested. If the delay is longer, however, it is determined that the connection path is more congested than before. Therefore, when the average queueing delay $d$ increases, $\alpha$ is reduced to a small value, leading to slower window increase. Likewise, the increased delay causes $\beta$ to be increased, so that the window reduction becomes greater.
The graph shows the shapes of $\alpha$ and $\beta$ functions as defined by the authors of the TCP Illinois algorithm [24], but important values are given as they are used by the Linux kernel. We use these values in subsequent experiments. First, the minimum value of $\alpha$ in Illinois is $\alpha_{\text{min}} = 0.3$ for the delay over $d_m$, and the maximum is $\alpha_{\text{max}} = 10$ for the delay less than $d_1$. However, in most network conditions, TCP Illinois has $\alpha > 1.0$. Namely, in the window growth, TCP Illinois is more aggressive than TCP NewReno because the latter always maintains $\alpha = 1$. On the other hand, in the highly congested network, $\alpha < 1.0$, so that it becomes even more conservative than NewReno. In the case of $\beta$, Illinois is almost always more aggressive than NewReno. Namely, it does not reduce the window size as quickly as NewReno except when the delay is very large, when $\beta_{\text{max}} = 0.5$ as in NewReno. The reduction can be as small as $\beta_{\text{min}} = 0.125$.

4.1.2. Modified TCP Illinois

Although exhibiting the best performance for the TCP connection traversing the given PLC link, TCP Illinois has not been specifically designed to cope better with lossy links. Therefore, similar to many other TCP variants, it also interprets and reacts to the packet losses as the network congestion signal. In this case, if packet loss occurs due to severe noise and load fluctuation when the electric car starts or ends charging, or when the electric car is turned on or off, it may unnecessarily slow down the transfer rate. Here, we modify the $\alpha$ and $\beta$ functions of TCP Illinois to not lose its aggressiveness for the bursty packet losses on the PLC link.

Liu et al. [24] considered the congestion window dynamics in Illinois compared to NewReno if packet losses are caused by the link quality, not by the network congestion. Let $W_I$ and $W_R$ be the average congestion window sizes of Illinois and NewReno, respectively. Then, their ratio is given by [24]:

$$\frac{W_I}{W_R} \approx \sqrt{\frac{\alpha}{2\beta}}$$

(4)

Note that the average window size is strongly related with the throughput. Because the throughput is the ratio of the window size to the round-trip time, and the round-trip time is the same for the two variants except for the queueing delay, the window size approximately equates to the throughput. Therefore, Equation (4) signifies the performance ratio of TCP Illinois to TCP NewReno.

There are two subcases when the queueing delay is very low. First, it may signal that there is no imminent congestion at the bottleneck. However, in this case, the losses will be infrequent. Second, the queueing delay is low but the loss rate is non-negligible. This second case is likely on a V2G communication path that involves a PLC link. In the case the PLC link is causing significant losses due to noise, the congestion window growth will be frequently stunted. The average queueing delay will be small, but the loss rate will be high. Based on Figure 9, we can conjecture that

$$\frac{W_I}{W_R} \approx \sqrt{\frac{\alpha_{\text{max}}}{2\beta_{\text{min}}}}$$

(5)
since \( \alpha \approx \alpha_{\text{max}} \) and \( \beta \approx \beta_{\text{max}} \) in such condition. If we can detect this condition, we can consider replacing \( \alpha_{\text{max}} \) and \( \beta_{\text{min}} \) so that the ratio \( \frac{W}{R} \) can be improved. Since TCP Illinois is a hybrid scheme, it can use both the delay information and the loss events to make it possible. Specifically, we can increase \( \alpha_{\text{max}} \) and decrease \( \beta_{\text{min}} \). For the V2G connection that includes the PLC link, therefore, we propose to change the functions for \( \alpha \) and \( \beta \) as in Figure 10.

![Figure 10](image)

**Figure 10.** The values of \( \alpha \) and \( \beta \) according to the \( d_a \) value of modified Illinois.

To revise \( \alpha \), we first bump up \( \alpha_{\text{max}} \) for \( d < d_1 \), where the congestion is not imminent. We also increase \( \alpha \) at \( d = d_m \). This is to make the modified TCP Illinois no less aggressive than TCP NewReno under any latency. For \( \beta \), we lower it for \( d < d_1 \). Namely, we make the modified TCP Illinois more aggressive when the delay is small. For \( d > d_3 \), \( \beta \) is the same as the original TCP Illinois and TCP NewReno. We could lower \( \beta \) even further, but, when the link error rate is very low (e.g., 0% or 0.1%), our experience shows that the throughput performance becomes worse. Thus, we maintain the \( \beta \) value in the large round-trip delay range. Below, we call this modification TCP Illinois with Added Aggressiveness (AA).

### 4.2. Modifying TCP NewReno for V2G Communication

Although TCP Illinois performs the best in the given V2G communication environment, TCP NewReno is the standard algorithm that is required by the ISO 15118. Therefore, an alternative approach that better conforms to the standard would be minor changes to TCP NewReno algorithm. In fact, in Section 5, we show that this approach performs comparably or even better in some cases relative to TCP Illinois.

In RFC 3390 [34], the initial \( ssthresh \) value is set to

\[
\min (4 \times \text{MSS}, \max (2 \times \text{MSS}, 4380))
\]

In other words, it is set between 2 MSS and 4 MSS, unless 2 MSS is smaller than 4380 bytes. More recently, however, Google proposed [35,36] and is using 10 MSS as the initial \( ssthresh \) value. Moreover, Linux also is using 10 MSS as the initial \( ssthresh \) value. RFC 6928 reports that the increased window size improves the latency for connection bandwidths closer to worldwide average with a median speed of about 1.7 Mbps, or even less [35,36]. On the standard PLC link bandwidth of 10 Mbps for V2G communication, which is shared by a few concurrent TCP connections, it would be in this neighborhood as well.

Our proposed modification is inspired by these recent studies and practices, but the modification in Algorithm 1 is not about the initial \( ssthresh \) value, but the \( ssthresh \) value that is used throughout the entire TCP connection lifetime. Algorithm 1 shows the simple modification that we propose for the TCP NewReno algorithm. The only deviation from the TCP NewReno algorithm is on Line 2. The Slow Start Threshold (\( ssthresh \)) is lower-bounded by 10 MSS. The reason is that, when the PLC link errors
are frequent, TCP NewReno spends a few more RTTs to regrow the congestion window size to previous values. Since the loss detection in TCP NewReno is mostly made through the retransmission timer expiry, the Slow Start is more frequently performed. Therefore, the impact of the $ssthresh$ value is larger in TCP NewReno than in TCP Illinois.

Algorithm 1: Modified NewReno Fast Retransmit.

```
1 if invoked by Fast Retransmit then
2     \[ ssthresh = \max(\frac{\text{FlightSize}}{2}, 10 \cdot \text{MSS}) \]
3       // used to be $2 \cdot \text{MSS}$
4     cwnd = ssthresh + 3
5     cwnd++
6     if partial ACK then
7         stay in Fast Recovery
8         retransmit next lost packet
9     else if full ACK then
10        cwnd = ssthresh
11        exit Fast Recovery and invoke Congestion Avoidance
12 end
```

Note that the only change we made to the original TCP NewReno algorithm is a change of a single constant, from 2 MSS to 10 MSS. However, we see in the next section that this small change brings large throughput improvement to the TCP NewReno performance on the lossy PLC link. The FlightSize is defined in RFC 2581 to be the amount of data that has been sent but not yet acknowledged. In Algorithm 1, the $ssthresh$ is set to the larger of half the FlightSize and 10 MSS. The latter has been increased in our modification from the 2 MSS specified by the RFC 3782. RFC 3782 is the NewReno modification to Fast Recovery Algorithm, and it specifies the $ssthresh$ value upon invoking Fast Retransmit as

\[ ssthresh = \max(\text{FlightSize}/2, 2 \cdot \text{SMSS}) \]

RFC 6928 argues that an initial window larger than three segments increases the chances of a lost packet to be recovered through Fast Retransmit as opposed to a lengthy RTO [35,36]. We believe that increasing the $ssthresh$ after the Fast Recovery (Line 10 in Algorithm 1) also increases this probability. For convenience, we call this modification TCP NewReno with Ssthresh Changed (SC) below.

5. Performance Comparison

In this section, we compared the performance of five TCP variants:

1. TCP NewReno
2. TCP NewReno with no Delayed ACK (NewReno ND)
3. TCP NewReno with proposed $ssthresh$ change (NewReno SC)
4. TCP Illinois
5. TCP Illinois with added aggressiveness (Illinois AA)

We include the TCP NewReno with the disabled Delayed ACK, because the Delayed ACK that is enabled by default slows the congestion window growth that happens on every ACK arrival at the sender.

We again perform two simulations with different numbers of TCP connections: 1 and 21. Figure 11a shows the results for the single TCP connection case. We first observe that TCP NewReno shows the worst throughput performance. Then, we notice that disabling the Delayed ACK algorithm improves upon the NewReno algorithm with Delayed ACK. It confirms that the faster growth, namely more aggressive congestion window sizing, pays off in the given V2G communication environment. We see
that TCP Illinois, as confirmed above, far outperforms the previous two TCP NewReno instances. The performance gap further widens when we introduce more aggressiveness to TCP Illinois by changing the $\alpha$ and $\beta$ values as discussed in the previous section. When the average packet loss rate is the highest, at 5%, modified Illinois has about 41% better throughput performance than existing Illinois.

Figure 11. Illinois, Illinois AA, NewReno, NewReno ND and NewReno SC performance comparison depending on the average packet loss rate of the PLC link with multiple electric vehicles.

What is most surprising in this result, however, is that the TCP NewReno algorithm with a single constant change in the $ssthresh$ size is significantly better than the two TCP NewReno siblings, except when the average packet loss rate on the PLC link is 0.25% or lower. It seems to imply that increasing the $ssthresh$ value, hence the aggressiveness, of the TCP NewReno algorithm slightly undermines the throughput when the PLC link losses are not severe. However, this performance reversal is minor compared to the big return that the proposed modification to TCP NewReno generates for larger packet loss rates. The throughput even exceeds that of TCP Illinois when the packet loss rate is 3.75% or higher. Although TCP NewReno is much simpler than TCP Illinois, the proposed change makes it fare comparably with the more complex counterpart.

Figure 11b shows the results of the experiment with 21 vehicles. What we see for the single TCP connection scenario above is mostly reiterated here. Minor changes are that, when the PLC link quality is good, the proposed NewReno modification with $ssthresh$ modification outperforms the other two NewReno implementations. Compared with the standard NewReno, the two Illinois variants and the modified NewReno significantly improve the throughput under all packet loss conditions. In particular, when there is no packet loss on the PLC link, the Illinois variants and the modified NewReno performs no worse than the standard NewReno. Namely, adding more aggressiveness as
proposed in this paper does not adversely affect the TCP throughput performance when the cause of packet losses is congestion.

Figure 12 sheds more light on the above results in terms of the congestion window sizes in the standard NewReno and the alternatives that we propose. With the same round-trip time condition, the window size is a direct indication of the TCP throughput performance. After the initial perturbation, we observe in the figure that the modified Illinois has much larger window sizes than the original Illinois or the NewReno. What is most interesting is the NewReno with the ssthresh increase, which shows that it maintains the window size over 10 MSSs (i.e., 15 KB) except when there are retransmission timeouts and consequent Slow Start. In other words, if there is no congestion or the congestion is so mild that losses are found by the Fast Retransmit, the modified NewReno does not unnecessarily slow down. Why such logic leads to much better throughput than the standard NewReno the following. When the window size is big enough, the PLC link losses do not always lead to timeouts.

Figure 12. Illinois, Illinois AA, NewReno and NewReno SC’s congestion window size movements, for 21 connections.

6. Conclusions

In a connected car, the performance of TCP is important as it is directly linked to the user experience. V2G communications using a PLC link that generates bursty errors due to sudden interference can adversely affect the TCP performance. In the V2G communication standard, NewReno is the standard TCP congestion control algorithm. NewReno does not fit well with such V2G communication environment because it was designed for the traditional Internet environment with congestion losses. In this paper, we show that TCP Illinois may be the better choice than NewReno for the TCP congestion control algorithm in V2G communication environment. However, if we should use NewReno, we show that it can be significantly improved by a minor ssthresh change in its aggressiveness. We observe by simulating the expected charging station environment that using TCP Illinois can improve the TCP throughput by up to 2.4 times when the average packet loss rate is 5%. In the case TCP NewReno is modified to maintain a larger window size, it can also achieve comparable throughput improvement. Our work demonstrates that the user experience can be significantly improved in Internet access through V2G communication, by the change of TCP congestion control implementation.

Author Contributions: Writing—original draft, J.P. and H.K.; writing—review and editing, J.P., H.K. and J.-Y.C.; conceptualization, J.P. and H.K.; investigation, J.P. and H.K.; and project administration, H.K. and J.-Y.C.
Funding: This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. 2017M3C4A7083676).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwon, D.; Park, S.; Baek, S.; Malaiya, R.K.; Yoon, G.; Ryu, J.T. A study on development of the blind spot detection system for the IoT-based smart connected car. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–14 January 2018; pp. 1–4.
2. Specification of TCP/IP Stack; AUTOSAR CP Release 4.3.1; AUTOSAR: Munich, Germany, 2017.
3. Wireless Access in Vehicular Environments (WAVE)—Networking Services; IEEE Standard 1609.3-2016; IEEE: Piscataway, NJ, USA, 2016. doi:10.1109/IEEESTD.2016.7458115. [CrossRef]
4. Road Vehicles—Vehicle-To-Grid Communication Interface—Part 2: Network and Application Protocol Requirements; ISO 15118-2:2014; ISO: Geneva, Switzerland, 2014.
5. Henderson, T.; Floyd, S.; Gurtov, A.; Nishida, Y. RFC6582: The NewReno Modification to TCP’s Fast Recovery Algorithm; IETF Internet Standard; IETF: Fremont, CA, USA, 2012.
6. Floyd, S.; Henderson, T.; Gurtov, A. The NewReno Modification to TCP’s Fast Recovery Algorithm; Technical Report; IETF: Fremont, CA, USA, 2004.
7. Road Vehicles—Vehicle Grid Communication Interface—Part 1: General Information and Use-Case Definition; ISO 15118-1:2019; ISO: Geneva, Switzerland, 2019.
8. Road Vehicles—Vehicle Grid Communication Interface—Part 3: Physical and Data Link Layer Requirements; ISO 15118-3:2015; ISO: Geneva, Switzerland, 2019.
9. Narrowband Orthogonal Frequency Division Multiplexing Power Line Communication Transceivers for G3-PLC Networks; ITU-T G.9903; ITU: Geneva, Switzerland, 2012.
10. Latchman, H.; Katar, S.; Yonge, L.; Amarsingh, A. High speed multimedia and smart energy PLC applications based on adaptations of HomePlug AV. In Proceedings of the 2013 IEEE 17th International Symposium on Power Line Communications and Its Applications (ISPLC 2013), Johannesburg, South Africa, 4–27 March 2013; pp. 143–148.
11. Yoon, S.G.; Kang, S.G.; Jeong, S.; Nam, C. Priority inversion prevention scheme for PLC vehicle-to-grid communications under the hidden station problem. IEEE Trans. Smart Grid 2017, 9, 5887–5896. [CrossRef]
12. Zimmermann, M.; Dostert, K. Analysis and modeling of impulsive noise in broad-band powerline communications. IEEE Trans. Electromagn. Compat. 2002, 44, 249–258. [CrossRef]
13. Galli, S.; Scaglione, A.; Wang, Z. For the grid and through the grid: The role of power line communications in the smart grid. Proc. IEEE 2011, 99, 998–1027. [CrossRef]
14. Estevez, C.; Cespedes, S. Improving performance of TCP-based applications in power line communications for smart grids. In Proceedings of the 2015 7th IEEE Latin-American Conference on Communications (LATINCOM), Arequipa, Peru, 4–6 November 2015; pp. 1–5.
15. Kumar, A. Comparative performance analysis of versions of TCP in a local network with a lossy link. IEEE/ACM Trans. Netw. 1998, 6, 485–498.
16. Brakmo, L.S.; Peterson, L.L. TCP Vegas: End to end congestion avoidance on a global Internet. IEEE J. Sel. Areas Commun. 1995, 13, 1465–1480. [CrossRef]
17. Fu, C.P.; Liew, S.C. TCP Veno: TCP enhancement for transmission over wireless access networks. IEEE J. Sel. Areas Commun. 2003, 21, 216–228.
18. Leith, D.; Shorten, R. H-TCP: TCP for high-speed and long-distance networks. In Proceedings of the PFLDnet, Lemont, IL, USA, 16–17 February 2004; Volume 2004.
19. Floyd, S. HighSpeed TCP for Large Congestion Windows; Technical Report; IETF: Fremont, CA, USA, 2003.
20. Cai, C.; Siferrincti, R. TCP Hybla: A TCP enhancement for heterogeneous networks. Int. J. Satell. Commun. Netw. 2004, 22, 547–566. [CrossRef]
21. Xu, L.; Harfoush, K.; Rhee, I. Binary increase congestion control (BIC) for fast long-distance networks. In Proceedings of the IEEE INFOCOM, Hong Kong, China, 7–11 March 2004; Volume 4, pp. 2514–2524.
22. Kelly, T. Scalable TCP: Improving performance in highspeed wide area networks. ACM SIGCOMM Comput. Commun. Rev. 2003, 33, 83–91. [CrossRef]
23. Ha, S.; Rhee, I.; Xu, L. CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Oper. Syst. Rev.* 2008, 42, 64–74. [CrossRef]

24. Liu, S.; Başar, T.; Srikant, R. TCP-Illinois: A loss-and-delay-based congestion control algorithm for high-speed networks. *Perform. Eval.* 2008, 65, 417–440. [CrossRef]

25. Mascolo, S.; Casetti, C.; Gerla, M.; Sanadidi, M.Y.; Wang, R. TCP westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, 16–21 July 2001; pp. 287–297.

26. Baiocchi, A.; Castellani, A.P.; Vacirca, F. YeAH-TCP: Yet another highspeed TCP. In *Proceedings of the PFLDnet*, Los Angeles, CA, USA, 7–9 February 2007; Volume 7, pp. 37–42.

27. Caini, C.; Furrinelli, R.; Lacamera, D. Comparative performance evaluation of tcp variants on satellite environments. In *Proceedings of the 2009 IEEE International Conference on Communications*, Dresden, Germany, 14–18 June 2009; pp. 1–5.

28. Tsiknas, K.; Stamatelos, G. Comparative performance evaluation of TCP variants in WiMAX (and WLANs) network configurations. *J. Comput. Netw. Commun.* 2012, 2012, 806272. [CrossRef]

29. Sarolahti, P.; Kuznetsov, A. Congestion Control in Linux TCP. In *Proceedings of the USENIX Annual Technical Conference*, FREENIX Track, Monterey, CA, USA, 10–15 June 2002; pp. 49–62.

30. Lin, J.; Nassar, M.; Evans, B.L. Impulsive noise mitigation in powerline communications using sparse Bayesian learning. *IEEE J. Sel. Areas Commun.* 2013, 31, 1172–1183. [CrossRef]

31. Tesla. 1533 Supercharger Stations with 13,344 Superchargers. 2019. Available online: https://www.tesla.com/en_GB/supercharger%20 (accessed on 24 August 2019).

32. Jacobson, V. Congestion avoidance and control. In *Proceedings of the ACM SIGCOMM Computer Communication Review*, Stanford, CA, USA, 16–18 August 1988; pp. 314–329.

33. Chiu, D.M.; Jain, R. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *j-COMP-NET-ISDN* 1989, 17, 1–14. [CrossRef]

34. Allman, M.; Floyd, S.; Partridge, C. RFC3390: Increasing TCP’s Initial Window; IETF Internet Standard; IETF: Fremont, CA, USA, 2002.

35. Dukkipati, N.; Refice, T.; Cheng, Y.; Chu, J.; Herbert, T.; Agarwal, A.; Jain, A.; Sutin, N. An Argument for Increasing TCP’s Initial Congestion Window. *ACM SIGCOMM Comput. Commun. Rev.* 2010, 40, 26–33. [CrossRef]

36. Chu, J.; Cheng, Y.; Dukkipati, N.; Mathis, M. RFC6928: Increasing TCP’s Initial Window; IETF Internet Standard; IETF: Fremont, CA, USA, 2013.