Adapting Time Headway in Cooperative Adaptive Cruise Control to Network Reliability

Adil Alsuhaim, Anjan Rayamajhi, James Westall, and Jim Martin, *Member, IEEE*

***Abstract*—Wireless network impairment results in adverse effect on string stability of a Cooperative Adaptive Cruise Control (CACC) platoon that relies on the wireless network, causing disturbances in the platoon that may lead to a collision. Previous work in the literature proposed fallback strategies that utilizes on-board sensors to recover in case of wireless network impairment, those methods assumes a fixed time headway value. In this paper, we study the string stability of a one-vehicle look-ahead CACC platoon under different network loss scenarios, and propose to adapt the time headway parameter of the model according to a network reliability metric that we defined based on packet burst loss length to maximize traffic flow efficiency while maintaining a string-stable platoon. Our findings show that careful adjustment of headway value according to the wireless network reliability allows the platoon to maintain string-stable operation while maximizing traffic flow.**

***Index Terms*—Cooperative Adaptive Cruise Control (CACC), Vehicular Ad-hoc NETworks (VANET), Connected Vehicles, Dedicated Short-Range Communication (DSRC), traffic flow, simulation**

I. Introduction

RECENT technological developments lead to the advent of Vehicular Ad-Hoc Networks (VANETs). VANETs are a special type of Mobile Ad-Hoc Network (MANETs), in which network nodes are vehicles equipped with wireless communication devices. Wireless Access in Vehicular Environment (WAVE) was specified by IEEE 802.11p in 2010 to add Dedicated Short-Range Communication (DSRC) between vehicles in the 5.9 GHz band. Another wireless technology that is currently in use in VANETs is Cellular Vehicle-To-Everything (C-V2X), which is a standard by The 3rd Generation Partnership Project (3GPP) that supports device-to-device communication without needing an infrastructure. The nature of communication in VANETs is a flat broadcast network, where vehicles are concerned with the delivery of information to their immediate neighbors.

Copyright © 2021 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

Manuscript received November 10, 2020; revised March 12, 2021 and July 5, 2021; accepted August 10, 2021. Date of publication *To be determined*, 2021; date of current version September 29, 2021.

Adil Alsuhaim is with the School of Computing, Clemson University, SC 29634, USA, and also with the Information and Computer Science department, King Fahd University of Petroleum & Minerals (KFUPM), Dhahran 31261, Saudi Arabia (e-mail: aalsuha@clemson.edu)

Anjan Rayamajhi is with Leidos Inc and works at Turner-Fairbank Highway Research Center in McLean, VA 22101, USA. (email: anjan.rayamajhi@leidos.com)

James Westall and Jim Martin are with the School of Computing, Clemson University, Clemson, SC 29634, USA. (email: westall@clemson.edu; jmarty@clemson.edu)

Digital Object Identifier XX.XXXX/TVT.2021.XXXXXXX

VANETs have attracted significant research interest from academicians and corporations, with a particular focus on Cooperative Adaptive Cruise Control (CACC) applications due to its potential in improving traffic flow, and therefore reduction in traffic congestion, fuel costs and emissions. Automobile manufacturers have been recently offering drivers the ability to engage an Adaptive Cruise Control (ACC) mode. In a traditional cruise control driving, a driver sets a desired speed, and the vehicle maintains that speed. In a vehicle equipped with ACC, the driver would set a desired cruise speed, and the vehicle attempts to maintain that speed, but it also uses on-vehicle sensors to estimate distance to the preceding vehicle, and reduces speed to avoid a collision. The availability of wireless communication in vehicles can allow them to share information, such as acceleration, with one another. CACC leverages information shared by vehicles to achieve more optimal traffic flow by forming a platoon of vehicles on the road with smaller separation. CACC is particularly interesting to transport companies because it can reduce the operating costs of trucks on the highways. DSRC-equipped vehicles do not typically have a centralized access coordinator. The IEEE 802.11 standard [1] specifies the process by which channel access is regulated. The standard specifies a contention-based medium access control procedure called "carrier sense multiple access with collision avoidance" (CSMA-CA). Since this process is somewhat random, it is possible that two vehicles might attempt to transmit a data frame at the same time, resulting in loss of data due to frame collision. This channel access mechanism offers no performance guarantees, or bounded delays. Vehicular applications such as CACC involve vehicles broadcasting periodic beacon messages with relevant information, and broadcast frames do not get acknowledgment of reception to minimize channel congestion. When the number of vehicles in the vicinity increases, the probability of packet loss due to wireless frame collision increases. In addition, it is possible for a malicious actor to perform a denial-of-service attack, rendering the channel useless. Failure of delivery of data can have adverse effects on the performance and safety of vehicular applications, and therefore it is important to study these applications in light of network condition.

CACC applications aim to improve *traffic flow*, which can be defined as the number of vehicles passing through a point on the road per unit of time. A CACC application can be parameterized with a desired time headway value between vehicles. A higher time headway means higher spacing margin at the expense of improved traffic flow. We would like to have this value as low as possible, while maintaining a string-stable

platoon operation.

In this work, we study the effect of network degradation on CACC operation, propose a metric for measuring the communication network reliability for CACC application, and use that metric to adjust the time headway value according to that metric to maximize traffic flow while maintaining a string-stable platoon operation. Under the loss processes studied, our time headway adaptation method achieves significant improvement in traffic flow over solutions that uses a fixed minimum safe headway, while also maintaining string stability.

II. Background

Longitudinal control of a platoon of vehicles has been extensively reported upon in the literature in the past decades. The objective of CACC is to set a spacing policy between vehicles of a CACC platoon using on-board sensors and a wireless communication network. The on-board sensors can measure distance from another vehicle, and the observed relative velocity. Wireless communication is used to communicate information between vehicles, and it can follow various topologies. Early designs of CACC models such as [2] suggest a bidirectional communication topology, where each vehicle makes use of information receives from its direct preceding and following vehicles. However, more recent work, such as California's PATH program [3], suggests that the use of a bidirectional information flow topology is unnecessarily complicated, and simplified the problem with a unidirectional approach in which a vehicle within a platoon is only concerned with information received from preceding vehicles. With unidirectional information flow, several control models were proposed, such as multi-vehicle look ahead [4] where information received from multiple preceding vehicles are used in addition to information from the platoon leader. In [5], it was shown that a higher traffic throughput achieved by a reduced time headway is possible by only using information from two preceding vehicles, without the need of information from the platoon leader. In [6] and [7], a CACC controller using a one-vehicle look-ahead were used, showing that it is possible to reduce the time headway gap well below 1 seconds while maintaining string stability. The work in [8] focused on string stability where the proportional-derivative parameters K_p and K_d were chosen for improved string stability over the choice in [6].

The technology used for vehicular communication, DSRC, does not offer performance guarantee, and therefore, attention was given to studying CACC under different network conditions. CACC makes use of acceleration information received from the preceding vehicle over the communication network. The work in [9] proposes that, in case of network impairment, a CACC application shall fallback to a degraded CACC state (dCACC), in which vehicles estimate the acceleration of preceding vehicles using only on-board sensors and using a Kalman filter to improve the accuracy of the estimation since sensors have an inherent measurement noise. It shows significant improvement over fallback to a traditional ACC as it dampens the disturbances better than ACC. However, their work does not adapt the time headway in the cases of a degraded network, and does not achieve string stability in the degraded state.

The term "string stability" refers to the resistance of a platoon to the propagation of disturbances from the front of a platoon to the back. A string-stable platoon is resilient to oscillating behavior of the platoon leader and was described in [2]. String stability of a platoon can be evaluated based on whether the platoon is homogeneous or heterogeneous [10]. A homogeneous platoon is a string of vehicles that are identical, with identical dynamics. Stability classically was evaluated based on the distance errors that the controller aims to minimize [11]. However, recent work bases string stability evaluation on vehicles' absolute positions [12], and velocities [13]. The impact of packet loss and different beaconing interval on the string stability of a platoon is studied in [14]. The study in [15] proposes a CACC system where vehicles also communicate only with their direct followers, but also takes into account heterogeneous vehicle dynamics such as vehicle length and braking performance. In [16], CACC is studied where applications mitigate communication delays by lowering network load to maintain string stability. Our work in [17] and [18] studies the impact of DSRC on the performance of CACC.

It is important to understand the nature of packet loss events exhibited by a VANET. This allows us to create realistic packet loss scenarios that CACC applications may encounter. In [19], the authors characterized packet loss events in 802.11p/WAVE vehicular networks that use multi-channel operations to derive a function to predict loss. They found that due to channel switching, packets might be dropped following a repeated pattern. In [20], the 802.11p communication channel is modeled based on Discrete Time Markov Chain (DTMC), where packet loss status is modeled with state machines of good states where packet reception is uninterrupted and loss states where packet loss is expected to occur with high probability. In [21], packet loss in VANETs is assumed to be bursty, and use a two-state Markov model to predict probability of packet loss based on Signal-to-noise ratio. For VANETs that use cellular-V2X communication, the work in [22] models packet losses in vehicular networks based on Pseudo-Markov chains and validated using a network that uses LTE. Packet delivery ratio (PDR) is less relevant in VANETs than inter-reception times [23], which correlates with the impact of burst losses that introduces increased delays between two successful receptions.

There is a consensus in the literature that packet loss occurrences are not independent. That is, they do not follow a Bernoulli process in which packets would be dropped with a certain probability, but should rather be expressed in a way similar to the Gilbert–Elliott model. In VANETs, packet loss events exhibit temporal and spacial correlation. Therefore, we focused our testing on losses of bursty nature rather than assigning an independent probability of loss to each packet.

In order to study network impairment and evaluate the fallback method's performance, we implemented a simulation based one a one-vehicle look-ahead controller. Since distance and relative velocity can be obtained using on-board sensors in case of network impairment, acceleration of the preceding vehicle can be estimated using a Kalman filter. In our study,

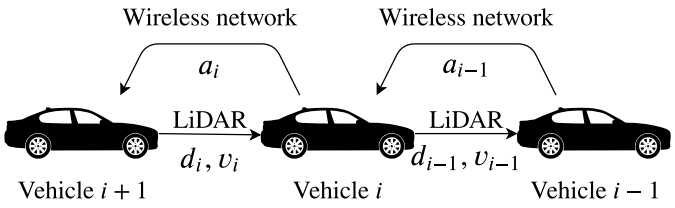


Fig. 1. One-Vehicle Look Ahead CACC

we simulated several packet loss scenarios and their effect on a CACC string stability. The main contribution of this work is the proposal of an algorithm that adapts the time headway value parameter of the CACC controller according to a network reliability metric we defined. Our results show that our algorithm allows the platoon to perform time headway adaptation while maintaining string stability throughout entire simulation runs for the tests we performed, achieving significant improvement over [9], which presented a fallback mechanism that does not maintain string stability, but only dampens disturbances compared to a fallback to conventional ACC.

VANETs can utilize different radio access technologies such as 802.11-based DSRC and Cellular Vehicle-to-Everything (C-V2X), and amendments are proposed to enhance the standards for use in VANETs, such as the P802.11bd that adds enhancement for next generation V2X. Our proposed solution can be applied to any radio access technology that can achieve the required low latency for CACC. The rest of this paper is outlined as follows: Section 3 provides a detailed description of the CACC control model, the criteria used for string stability and fallback mitigation when network connection is lost. Section 4 details our simulation setup and the time headway adaption process based on network reliability. Section 5 is a discussion of the results from simulation.

III. CACC Controller Model & Mitigation

A CACC controller requires the presence of a wireless communication network over which vehicles can broadcast information periodically. This information can include acceleration, velocity and position. However, if vehicles rely solely on information received over a wireless network, there is no fallback strategy if network becomes unavailable. Therefore, vehicles can use local on-board sensors such as LiDAR to detect the distance and relative velocity of the preceding vehicle, and estimate its acceleration

A. CACC Controller Model

We implemented an ns-3 simulation of a one-vehicle look-ahead CACC model as shown in Figure 1. The CACC model used is based on [6]. If we have a CACC platoon of N vehicles, a spacing policy for vehicle i can be defined with

$$d_{r,i}(t) = r_i + hv_i(t), \quad 2 \leq i \leq N \tag{1}$$

where h is the desired time headway in seconds. r_i is the stand-still reference distance in meters. The spacing error between the desired and actual spacing between vehicles i and $i-1$ is denoted by $e_i(t)$, and computed as follows:

$$\begin{aligned} e_i(t) &= d_i(t) - d_{r,i}(t) \\ &= (s_{i-1}(t) - s_i(t) - L_i) - (r_i + hv_i(t)) \end{aligned} \tag{2}$$

where $s_i(t)$ is the absolute position of vehicle i at time t, and L_i is the length of vehicle i.

The CACC controller for a following vehicle i computes a new acceleration every t_{CACC} period of 100ms based on the following equation:

$$\dot{u}_i = -\frac{1}{h}a_i + \frac{1}{h}\left(k_p e_{1,i} + k_d e_{2,i}\right) + \frac{1}{h}u_{i-1} \tag{3}$$

where a_i is the the current vehicle's acceleration, $e_{2,i}$ is the change in $e_{1,i}$, namely $\dot{e}_{1,i}$, and k_p, k_d are gain parameters set to 0.66 and 0.7 respectively. The value of u_{i-1} is the perceived acceleration of the preceding vehicle, which is either received over wireless communication or estimated from on-board sensor reading when the network is impaired. The value $\dot{u}$ is the change in acceleration from a starting acceleration a_{start} to a target acceleration a_{target}, which is performed in a linear manner over a CACC update interval t_{CACC}. For example, at time t_{start}, vehicle i computes a new target acceleration a_{target}, and changes its current acceleration linearly until it reaches the target acceleration at $t_{\text{target}} = t_{\text{start}} + t_{\text{CACC}}$, where a new target acceleration would be computed again, and so on. The value u_{i-1} that vehicle $i-1$ broadcasts over the wireless network can either be its current acceleration, or its target acceleration computed from Equation 3

Vehicles' absolute positions and velocities are updated accordingly as well, following a piecewise acceleration whose slope j_i is computed, every time a new acceleration target is computed, as:

$$j_i = \frac{a_{\text{target}} - a_{\text{start}}}{t_{\text{CACC}}} \tag{4}$$

and then at a time $t \in [t_{\text{start}}, t_{\text{target}}]$, vehicle i's acceleration $a_i(t)$, velocity $v_i(t)$ and position $d_i(t)$ are updated continuously according to the following motion equations:

$$a_i(t) = a_i(t_{\text{start}}) + j_i \Delta t \tag{5}$$

$$v_i(t) = v_i(t_{\text{start}}) + a_i(t)\Delta t + j_i \frac{\Delta t^2}{2} \tag{6}$$

$$d_i(t) = d_i(t_{\text{start}}) + v_i(t)\Delta t + a_i(t)\frac{\Delta t^2}{2} + j_i \frac{\Delta t^3}{6} \tag{7}$$

where Δt is $t - t_{\text{start}}$. The platoon leader follows an acceleration profile that is read from a text file, where the leader starts from a velocity and acceleration of 0, and obtains a new target acceleration from the file every 100ms, following the aforementioned motion equations.

B. String Stability

In order to evaluate the performance of a CACC deployment, we need to determine whether the platoon is *string stable*. For a string-stable platoon, the amplification of disturbances in upstream direction of either distance error, velocity or acceleration is diminished before they reach the end of the platoon. In this section we review string stability from the

perspective of [6], [24]. In Section II of both [6] and [24] the authors present a brief discussion of the concepts of string stability. They identify three general approaches that have been applied. The first is a formal approach based upon Lyapunov stability. This approach is said to suffer from constraints that make it lack practicality in real-world systems. Specifically, external disturbances, including speed variation of the lead vehicle, are said to be not supported. In a second approach, the Z-transform is used in analysis of infinite strings. The authors point out in [24] that "Unfortunately, the stability properties of finite-length strings, being practically relevant, might not converge to those of infinite-length strings as length increases. This can be understood intuitively by recognizing that in a finite-length platoon, there will always be a first and a last vehicle, whose dynamics may significantly differ from those of the other vehicles in the platoon, depending on the controller topology. Consequently, the infinite-length platoon model does not always serve as a useful paradigm for a finite-length platoon as it becomes increasingly long."

The third approach is referred to as a *performance-oriented* approach. A finite string of N vehicles is *defined* to be string stable if and only if

$$||z_i(t)||_{\mathcal{L}_2} \leq ||z_{i-1}(t)||_{\mathcal{L}_2} \ \ \forall t \geq 0 \ \text{ and } \ 2 \leq i \leq N \tag{8}$$

where $z_i(t)$ can be (with equivalent results) the positional error $e_i(t)$ of vehicle i with respect to vehicle $i-1$, its speed $v_i(t)$, or its acceleration $a_i(t)$.

This metric also includes an assumption that the initial state of the string is stationary. Consequently, it is not applicable to strings whose membership may dynamically change nor to analysis of sub-intervals of string behavior.

Nevertheless, for simulation studies, it is clear that this metric is readily applied to any finite length platoon with any combination of controller inputs and the required initial conditions present no obstacle. Therefore, in this work we also employ this specific characterization of string stability and use $v_i(t)$ in our computations.

This stability condition is a retrospective analysis of the platoon stability, and can be expressed as a single value that, for a stable platoon, must satisfy the condition:

$$\frac{||v_i(t)||_{\mathcal{L}_2}}{||v_{i-1}(t)||_{\mathcal{L}_2}} \leq 1 \tag{9}$$

C. Fallback mitigation

The CACC controller uses periodic acceleration updates received over the wireless network. When the network becomes unavailable, these updates are lost, and therefore the model does not receive updated acceleration inputs. The CACC controller in each vehicle computes a new target acceleration every 100ms, and it also expects to receive acceleration updates from the preceding vehicle every 100ms. If acceleration updates were not received, the controller will use the last known acceleration value until it detects packet loss. In VANET based on DSRC/WAVE, packet loss is not explicitly detected, and it is possible to occasionally miss some acceleration updates due to frame collisions. The controller will use the last known acceleration from preceding vehicle as long as the last update received was less than 300ms ago. If a vehicle does not receive an acceleration update after 300ms, it assumes that the communication network is in a loss state, and switches to a fallback mode, where acceleration of the preceding vehicle is estimated using on-board sensors.

The fallback mode works in the absence of the wireless network by using on-board sensors and a Kalman filter to estimate the preceding vehicle acceleration as detailed in the work in [18]. In ns-3, we can obtain the exact distance and relative velocity between two simulation nodes. However, this is unrealistic as real-world sensors have inaccuracies within a certain margin of error. Therefore, we applied a zero-mean Gaussian noise to the actual distance between simulation nodes, such that distance error is within ± 0.3 meters, and speed accuracy is within +2 km/h and -3 km/h. These margins of errors are based on the accuracy requirement for approval by National Highway Traffic Safety Administration (NHTSA) [25]. We assume that the variance in distance error σ_d^2 and the variance in relative velocity error σ_v^2 to be 0.1 in our simulation. We will refer to the added distance and relative velocity sensor noises as $\mathcal{N}_{dist}$ and $\mathcal{N}_{vel}$ respectively.

A Kalman filter has two sub-processes executing recursively in a loop: *prediction phase* which periodically predicts the state variables, and correction phase which attempts to correct the predicted state variable values using the measurement. If x is the state of a system to be estimated, u is control input, and z is the measurements done periodically, then a Kalman filter is designed to address the general problem of estimating the state $x_k \in \mathcal{R}^n$ with measurements $z \in \mathcal{R}^m$ of a process defines by the discrete time equations:

$$x_k = \mathbf{A}x_{k-1} + \mathbf{B}u_{k-1} + w_{k-1} \tag{10}$$

$$z_k = \mathbf{H}x_k + v_k \tag{11}$$

The random variables w_k an v_k are assumed independent zero-mean Gaussian random variables that represent process and measurement noise respectively. The covariance matrices of the process noise and measurement noise are $\mathbf{Q}$ and $\mathbf{R}$, which are determined for a sensor in practice by trial and error for smaller mean squared error. The $n \times n$ matrix $\mathbf{A}$ relates the past estimate of the state x_{k-1} to the current x_k in the absence of noise and control input u_k. Similarly, the $n \times 1$ matrix $\mathbf{B}$ relates the control input u_k to the state x_k. The $m \times n$ matrix $\mathbf{H}$ relates the state x_k to the measurement variables z_k. In practice, the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{H}$ vary with time but we assume they remain constant for our study.

We also initialize $\mathbf{P}$ as a 4x4 matrix with all elements set to 1. The Kalman filter's prediction phase:

$$\hat{x}_k^- = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}\hat{u}_{k-1} \tag{12}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{13}$$

where $\hat{x}_k^-$ is the priori estimate of the state x_k at time step k and $\hat{x}_k$ is the posteriori estimate of state x_k at time step k. $\mathbf{P}_k^-$ is the priori estimate error covariance and and $\mathbf{P}_{k-1}$ is the posteriori estimate error covariance. The goal is to find an equation that computes a posteriori estimate $\hat{x}_k$ as a linear combination of an a priori estimate $\hat{x}_k^-$ and a weighted

difference between an actual measurement z_k and a prediction $\mathbf{H}\hat{x}_k^-$. The Kalman *correction phase* is defined by:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^t + \mathbf{R})^{-1} \quad (14)$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \quad (15)$$

$$\hat{x}_k = \hat{x}_k^- + \mathbf{K}_k(z_k - \mathbf{H}\hat{x}_k^-) \quad (16)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- \quad (17)$$

where the $n \times n$ matrix $\mathbf{K}_k$ is the gain that minimizes a priori error covariance $\mathbf{P}_k^-$ at time step k.

The relative distance from vehicle $i-1$ to preceding vehicle i at time k is computed with the addition of sensor noise as follows:

$$d_i[k] = x_{i-1}[k] - x_i[k] + \mathcal{N}_{dist} \quad (18)$$

Similarly, the relative velocity between vehicles $i-1$ and i at time k is computed by adding the noise value for speed measurement as follows:

$$r_i[k] = v_{i-1}[k] - v_i[k] + \mathcal{N}_{vel} \quad (19)$$

Between two consecutive sensor readings, we will define distance traveled by vehicle i as $s_i[k]$, and using two consecutive sensor readings, we can estimate the distance traveled by preceding vehicle $i-1$ as

$$\hat{s}_{i-1}[k] = s_i[k] + d_i[k] - \hat{s}_{i-1}[k-1] \quad (20)$$

where $\hat{s}_{i-1}[k-1]$ is the prior estimate of distance traveled by the leading vehicle i-1. The velocity of preceding vehicle $i-1$ is estimated from the relative velocity reading and the velocity of vehicle i as follows

$$\hat{v}_{i-1}[k] = r_i[k] + v_i[k] \quad (21)$$

The acceleration of the preceding vehicle $i-1$ is estimated using the change in estimated velocity between two sensor readings. Let t_{dme} be the time between two sensor readings, acceleration is estimated as:

$$\hat{a}_{i-1}[k] = \frac{\hat{v}_{i-1}[k] - \hat{v}_{i-1}[k-1]}{t_{\text{dme}}} \quad (22)$$

We will use the term $\hat{m}_{i-1}[k]$ to denote the approximate rate of change of preceding vehicle's acceleration at time k, and therefore, we establish the state variable $x[k]$ as:

$$x[k] = \begin{bmatrix} m_{i-1}[k] \\ s_{i-1}[k] \\ v_{i-1}[k] \\ a_{i-1}[k] \end{bmatrix} \begin{matrix} \text{Denotes change in acceleration, or } \textit{jerk} \\ \text{Denotes } \textit{distance traveled} \\ \text{Denotes } \textit{velocity} \\ \text{Denotes } \textit{acceleration} \end{matrix} \quad (23)$$

The Kalman filter model parameters at any time step k are given by defining the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{H}$ and u as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{t_{\text{dme}}} \\ \frac{t_{\text{dme}}^3}{6} & 0 & t_{\text{dme}} & \frac{t_{\text{dme}}^2}{2.0} \\ \frac{t_{\text{dme}}^2}{2.0} & 0 & 1 & t_{\text{dme}} \\ t_{\text{dme}} & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad u = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

We also defined the measured estimated state of preceding vehicle $i-1$, denoted with z as follows:

$$z = \begin{bmatrix} \hat{m}_{i-1}[k] \\ \hat{s}_{i-1}[k] \\ \hat{v}_{i-1}[k] \\ \hat{a}_{i-1}[k] \end{bmatrix}$$

We use two covariance matrices $\mathbf{Q}$ and $\mathbf{R}$, which in practice would be provided by a sensor's manufacturers, but since our study is simulation-based, we derive $\mathbf{Q}$ and $\mathbf{R}$ to be in line with the error variance of 0.1 as follows:

$$\mathbf{Q} = \begin{bmatrix} 0.1^{-4} & 0 & 0 & 0 \\ 0 & 0.1^{-5} & 0 & 0 \\ 0 & 0 & 0.1^{-5} & 0 \\ 0 & 0 & 0 & 0.1^{-5} \end{bmatrix},$$

$$\mathbf{R} = \begin{bmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

The estimated acceleration value of the preceding vehicle is extracted from the $\hat{x}_k$ vector, and is always calculated with every sensor reading. If a vehicle missed three consecutive acceleration updates through the wireless network, it will use the most recent acceleration estimated by the Kalman filter described here. Without using a Kalman filter, it is still possible to estimate the acceleration of the preceding vehicle, however our tests show that using a Kalman filter produces smaller error in estimation as shown in Figure 2.

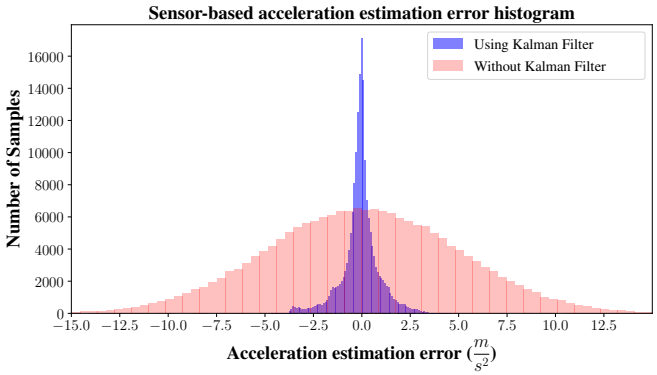


Fig. 2. Improvement in acceleration estimation using a Kalman filter

IV. Simulation Setup & Experiments

A. Simulation Setup

We use ns-3 network simulator to create a one-vehicle look-ahead CACC application. We created 10 vehicles: a platoon leader and nine followers. The simulation starts with stationary nodes separated by a given reference distance. We assume a homogeneous platoon of vehicles with identical dynamics, and length of 4.5 meters as shown in Table I. The platoon leader

TABLE I
NS-3 SIMULATION PARAMETERS

Beacon Interval	100 ms
Sensor interval	25ms
Number of vehicles	10 vehicles
Vehicle length	4.5 meters
Velocity range	0 to 35 m/s
Acceleration range	-3 to 3 m/s^2
K_p	0.66
K_d	0.7

is controlled directly by feeding it new acceleration values every 100ms from a preset acceleration profile. This simulates a real vehicle driven by a human driver, or an autonomous self-driving vehicle. The remaining vehicles in the platoon calculate their acceleration based on data received from the vehicle directly in front of them and their on-board sensors. We can get exact distance and relative velocity between two simulation nodes in ns-3, but we add noise to this data to simulate error in sensor readings. To study the effect of burst loss effect on the platoon's string stability, we used synthetic acceleration profiles that follow linear, sinusoidal, and step functions. We also used two profiles obtained from real vehicles, namely the *us06* drive cycle [26], and *c2a*, a profile we generated by driving a car from Clemson, SC to Anderson, SC. To study our proposed solution, we needed longer drive cycle data, so we used five acceleration profiles obtained from real vehicles that we acquired from the National Renewable Energy Laboratory (NREL) [27].

We implemented a simulation with a controlled communication channel that can be configured with the desired loss process and the one-way latency of the delivery of packets. The results presented reflect simulation runs with a one-way latency of 750 μs. We also re-ran the simulations with a one-way latency of up to 20ms with no significant difference in the final results.

A simulation stops if a collision occurred or when the specified total simulation time is reached. After a simulation concludes, the observed string stability and traffic flow is computed. To compute traffic flow for a platoon of N vehicles, let $x_1(t)$ and $x_N(t)$ be the positions of the platoon leader and the N-th vehicle at time t respectively. Let t_e be the time at which the N-th vehicle passes the point $x_1(t)$. Traffic flow $\mathcal{F}(t_e)$ at time t_e can be defined as:

$$\mathcal{F}(t_e) = \frac{N}{t_e - t} \quad \text{vehicles per second} \tag{24}$$

We repeat traffic flow computation every time the N-th vehicle passes a position where the platoon leader once was until the end of a simulation run. Simulation results are acceptable only if they resulted in a string-stable platoon, and did not cause a collision. It is common to express traffic flow in *vehicles per hour*, and therefore we will multiply the result by 3600.

B. Network Reliability Assessment & Time Headway Value

We studied how CACC platoons react to different packet loss processes. We noticed that when modeling loss events as a Bernoulli process, the CACC is resilient even with high packet loss ratios. However, since previous work provided solid evidence that packet losses are bursty, we needed to test how would CACC performs under these conditions. As would be expected, CACC performance is influenced by the length of packet loss bursts. Smaller burst losses of 10 packets (lasting for 1 second) do not impact the stability of a CACC platoon, allowing us to set a lower time headway value h. As we tested with longer burst losses, the platoons become unstable, requiring the use of higher values of h to maintain stability. We call the value of h that ensures string-stable operation under a given loss process the *minimum string-stable time headway*. When burst length goes above 300 packets (30 seconds), the minimum string-stable time headway value remains more or less constant as time headway becomes sufficient for the fallback method to work without causing instability as shown in Figure 3.

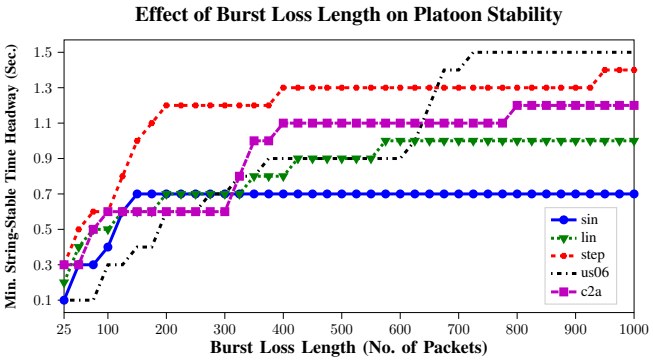


Fig. 3. Packet burst length effect on platoon stability for different acceleration profiles

We seek to assess the network reliability to base our decision of adapting the h value to achieve string-stable platoon operation and maximize traffic flow. Packet delivery ratio (PDR) would be feasible if the loss process followed a Bernoulli process. However, we needed to devise a mechanism based on the bursts of packet losses. We devised a network reliability metric, which corresponds to the effect of long loss lengths on the minimum string-stable time headway value as follows

$$r(L_B) = 1 - \tanh\left(\frac{L_B \times M}{175}\right) \tag{25}$$

where L_B is the length of the maximum sustained burst of packet loss that the network experienced, $\tanh$ is the hyperbolic tangent function, and M is a scaling multiplier that our algorithm can control to make the criteria more or less conservative in estimating reliability. The default value of M is 1, but can change during simulation. The choice of the $\tanh$ function in this manner provides a graph that is inversely proportional to the effect burst of packet losses, and therefore a good fit to assess reliability as it pertains to CACC platoon stability.

Vehicles of the CACC platoon start in a *no loss* state, where all packets are received as expected every 100ms. If a loss is detected, a vehicle will enter into a *burst loss* state in which the maximum burst length L_B is maintained, which is used a function $f(L_B)$ to adapt the value of the time headway value

h. When packet reception resumes, we could have multiple scenarios about the reception after loss event :

1) the duration of loss free reception is very short, meaning that network reliability has not improved significantly. As a result, subsequent losses are added to the counter b, which counts consecutive lost packets whereas the counter g used for consecutive packet receptions is reset to 0.
2) the duration of loss free reception is moderate, so we will reset the counters b and g, meaning that subsequent losses are not added to the maximum observed burst loss L_B unless a loss greater than L_B is observed.
3) the duration of loss free reception is long enough to assume that the network condition has improved, so we move to a *recovery state*.

This is illustrated in the state-machine in Figure 4, where p and q are used as constant parameters to define the thresholds that defines the transitions. In our simulation, we set $p = 2$ and $q = 5$. This means that we have to receive consecutive packets in a burst that is five times the size of the largest burst before we assume that the network is back to perfect condition. In addition, if good burst between losses is two times the loss length, we will set the multiplier M to 0.8, since we observed that this kind of loss is more forgiving on string stability. If good packet burst between loss bursts is shorter than 25% of L_B, we set the multiplier M to 4, and if it is less than 10%, we set the M to 20 as we have observed that short good bursts of packets between significantly longer burst losses have severe impact on the string stability of CACC operation. We use the value of L_B which we defined as the maximum burst loss length, to compute a new higher headway value to mitigate packet loss events as follows:

$$h = f(L_B) = (h_{\text{lower}} - h_{\text{upper}})r(L_B) + h_{\text{upper}} \qquad (26)$$

Where h_{lower} and h_{upper} are the lower and upper bounds on the value of the time headway value h. We notice that from Figure 3 that for all acceleration profiles that we tested, an upper bound h_{upper} of 1.5 seconds is sufficient with very long packet burst loss, that is, it can be used when there is a total network outage. We also limit the maximum value of L_B to 300 since that value will produce a network reliability of zero.

When we reach *recovery state*, the value of the time headway h is reduced. Our initial experiments determined that reducing h too quickly could result in instability or a collision. Therefore the process is performed gradually by δ over an update period t seconds. We choose a value of $\delta = 0.05$. In addition, we also reduce h in a front-to-back order, that is vehicles in the front of the platoon reduce reduce their time headway earlier than the vehicles in the back, and over shorter interval of time which we found helps maintain platoon stability.

C. Packet Loss Scenarios

Packet loss is modeled using a state-machine with two states, a *reception state* where all packets are received, and a *loss state* where packets are lost as shown in Figure 5. The

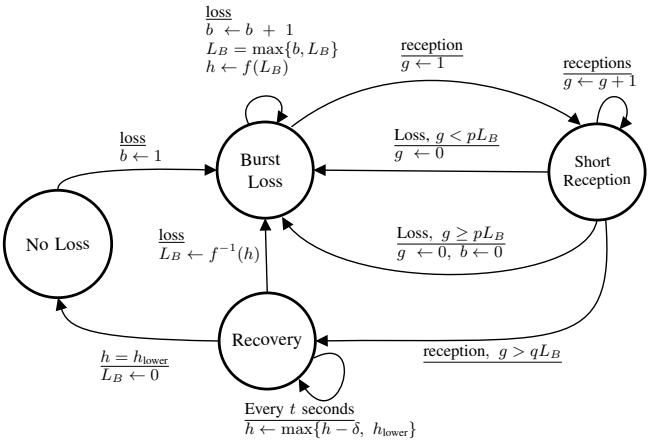


Fig. 4. State machine for evaluating network reliability for CACC.

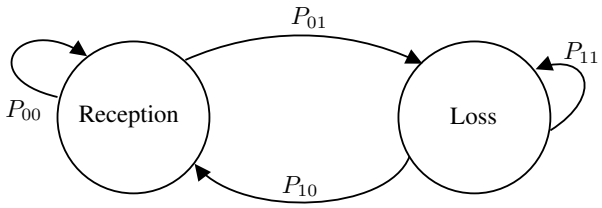


Fig. 5. Modeling packet loss scenarios

switching between a reception and loss state is stochastically controlled. If we use 0 and 1 denote the reception and loss states respectively, we can define P_{ij} as the probability to transition to state j when at state i. Those probabilities can be represented with a 2x2 matrix. We can parameterized the loss process with two parameters: the Mean Good Length (MGL) which represents the mean number of received packets in a reception state, and the Mean Burst Length (MBL) which represents the mean number of lost packets in a loss state, and we compute the values of $P_{00}, P_{01}, P_{11}, P_{10}$ as follows:

$$P_{00} = \frac{MGL - 1}{MGL}, \quad P_{01} = \frac{1}{MGL}, \qquad (27)$$

$$P_{11} = \frac{MBL - 1}{MBL}, \quad P_{10} = \frac{1}{MBL} \qquad (28)$$

We setup our CACC simulation to start with an ideal wireless network connectivity with no packet loss, and later we trigger a loss event during simulation to study the impact of different loss processes on the performance of CACC. We tested using three loss processes shown in Table II. A denial-of-service (DoS) attack can take an on-off pattern [28] so we modeled it as a loss process with $MGL = 50, MBL = 50$. The network outage scenario represents a condition where communication is lost completely due to temporary malfunction in equipment, or a complete denial-of-service (DoS) attack. To model loss in a congested network, we performed simulation test with many vehicles using the same communication channel and broadcasting every 100ms, and observed that loss takes patterns of longer bursts followed by shorter reception bursts.

In a simulation where there is no packet loss, the value of the time headway h can be set very low, allowing for higher vehicular traffic throughput. If we chose a very small h value,

and then triggered packet loss, it is possible for instability or a collision to occur.

Therefore, when packet loss occurs, we use the reliability metric discussed earlier to renew the value with a higher value of h, which effectively increases the separation distance between vehicles, which consequently reduces the risk of instability or collisions.

TABLE II
PACKET LOSS PROCESSES USED IN TESTING

Loss Process	MGL	MBL
On/Off Denial of Service	50	50
Network outage	1	10000
Congestion	10	40

The loss process can be applied to vehicles individually, and due to its stochastic nature, different vehicles would exhibit different loss lengths which would result in different reliability metric observed by different nodes, and hence different headway values chosen by the vehicles. To mitigate that, we make a simplifying assumption that every vehicle would broadcasts its current observed network reliability metric value along with the acceleration. Moreover, it is possible to assume that since the communication range of conventional DSRC devices reaches 1000 meters, all vehicles are able to receive frames intended for other vehicles, and therefore, can compute the lowest reliability observed by a member of the platoon. Alternatively, CACC vehicles can also use an alternative WAVE channel to broadcast reliability or use cellular LTE connection to share the network reliability value. This allows every platoon member to have a global view of reliability. It is possible to consider only local reliability metrics, but that requires more conservative network reliability to achieve string stability under loss.

D. Dynamic Adjustment of Headway Value

The idea of our work is to allow our CACC platoon to use the minimum string-stable time headway value when the network condition is perfect. Our tests with ten vehicles with platoon leader following drive cycles obtained from NREL show that the minimum string-stable time headway to use is $h = 0.5$. When the network condition is degraded, vehicles resort to estimating the acceleration of their respective preceding vehicle using on-board sensors which is not as accurate as receiving acceleration updates over the wireless network. This has the potential of increasing the risk of instability or collision with a low fixed headway value. Our tests with ten vehicles showed that for $h \geq 1.5$ we can achieve string-stable platoon operation with sensor-estimated acceleration alone, i.e. without a a wireless network.

Our headway adaptation algorithm sets the headway value such that $h \in [0.5, 1.5]$. In a trivial loss scenario where we run with a perfect network condition, followed by a period of total network outage before resuming perfect operation, one would assume that adjusting the value of h from 0.5 to 1.5 and then back to 0.5 would suffice as a countermeasure. However, rapid reduction of h after network condition resumes to a perfect condition can lead to instability or a collision.

Therefore, we opted to have the reduction of h value to be gradual, and in a front-to-back manner: the second vehicle in the platoon (the direct platoon leader's follower) initiates the reduction first, followed by the third vehicle and so on. This process allowed us to change the headway value without breaking string stability of the platoon. In addition, when network condition degrades, headway values can be increased in an opposite direction, starting from the platoon's tail to front. Equation 26 is used to determine a target h value when a loss occurs to increase the headway value, effectively increasing the separation between vehicles during less-than-ideal network condition. For every vehicle i, we increase the value of h to the newly computed h value gradually over a period of 2 seconds.

The algorithm we devised to reduce h value when the network is back to perfect condition is listed in Algorithm 1. We assume that every vehicle i knows its position within the platoon, which allows us to perform the reduction in a front-to-back manner. The inputs i, h, and δ correspond to the vehicle's index within the platoon, the current time headway value and the decrement by which we reduce the headway value. δ was set to 0.02 for a very careful gradual change to the time headway value to reach h_{new}. The value T_{update} represents the time it takes a vehicle to update to the target time headway h_{new}. Our idea relies on two concepts to minimize the chance of instability: the reduction of headway value should be gradual, and the reduction should start at the front of the platoon. This idea can be achieved by exchanging extra information in the messages to achieve the desired outcome. However, to reduce the complexity, we simply use a node's index to determine the delay until the initiation of the gradual reduction process, and the duration of the gradual reduction process. The reduction algorithm is invoked periodically every 10 seconds when the network condition is perfect after a loss process.

Algorithm 1: Reduction of h when network condition is perfect

Input: i, h, δ
Output: Value of h is reduced gradually for vehicle i
Function `HeadwayReduction`(i, h, δ)**:**
  while *Network is perfect* **do**
    $T_{\text{update}} \leftarrow 2 \times i$ seconds
    $h_{\text{new}} \leftarrow \max\{h - \delta, L\}$
    slope $\leftarrow \dfrac{h_{\text{new}} - h}{T_{\text{update}}}$
    delay $\leftarrow T_{\text{update}}$
    call `LinearReduction`(slope, h_{new}, delay)

Function `LinearReduction`$(slope, h_{new}, delay)$**:**
  Wait for delay seconds
  while $h \neq h_{new}$ **do**
    $h \leftarrow h + 0.1$slope
    Wait for 100ms

V. Results & Analysis

As mentioned earlier, we ran several experiments to decide on the lower and upper bounds for h since this is a necessary input to the dynamic headway assignment algorithms. The leader's acceleration profile samples shown in Table III can achieve string-stable platoon operation under no packet loss scenario using a minimum string-stable time headway value of 0.5. We also observed that a maximum headway value of 1.5 would result in string-stable operation under network outage scenario, and therefore our adaptation algorithm uses a value of $h \in [0.5, 1.5]$.

TABLE III
Acceleration Profiles Used for the Experiments

Acceleration Profile	Simulation Time	Source
Sample 1	1918 seconds	Austin, TX
Sample 2	4457 seconds	Atlanta, GA
Sample 3	2614 seconds	Austin, TX
Sample 4	2870 seconds	Austin, TX
Sample 5	2400 seconds	Caltrans, California

Acceptable results must have string stability and no collision. Therefore, our simulations compare the performance of using the minimum fixed time headway value of h versus our dynamic adjustment approach of the value of h based on network condition. The minimum string-stable time value of h varies with acceleration profile, and network loss process applied.

We used data obtained from NREL's website [27] that contains drive cycle data including acceleration, which is sampled every 1 second. Because our implementation of CACC relies on acceleration updates every 100ms, we used a simple linear interpolation to fill in the data points between every two acceleration to make it a per-100ms acceleration profile.

For every simulation run, we assess string stability according to Equation 9. In Figure 6, we illustrate the difference between a string-stable platoon running our adaptive method versus an unstable platoon that uses a fixed headway, both under the same loss process applied at $t = 500$, using the velocities of the last two vehicles in a platoon of 10 vehicles. We can see that our adaptation maintains string stability when network loss occurs.

A. Results for 50/50 Loss Process

For each acceleration profile in Table III, we run multiple CACC experiments with a constant headway time value h, and we applied a loss process of $MGL = 50, MBL = 50$ for 300 seconds (5 minutes), applied between simulation times 500 up to 800. The aim is to establish what is the minimum h that can be used without causing a collision or breaking string stability under the loss process. We then compare the traffic flow by that minimum string-stable h with the adaptive approach as shown in Table IV, where traffic flow is quantified in units of *vehicles per hour*. In Figure 7(a), we observe that the overall traffic flow is increased for the simulation run with Sample 1. The shaded area marks the time in which the loss process is applied. As we can see, the performance during the loss process matches the traffic flow of the scenario where

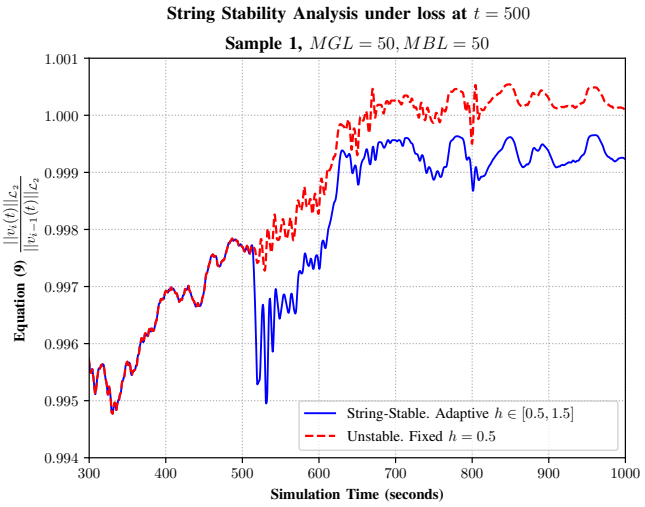


Fig. 6. String stability as per Equation 9

TABLE IV
Traffic Throughput (*vehicle/hour*) Achieved Under a 50/50 Loss Process With Different Acceleration Profiles.

Acceleration Profile	Fixed h	Adaptive h	Improvement %
Sample 1	2743.29	3385.45	+23.41%
Sample 2	2625.33	3284.89	+25.12%
Sample 3	2777.15	3208.74	+15.54%
Sample 4	2500.3	2807.37	+12.28%
Sample 5	2869.26	3286.39	+14.54%

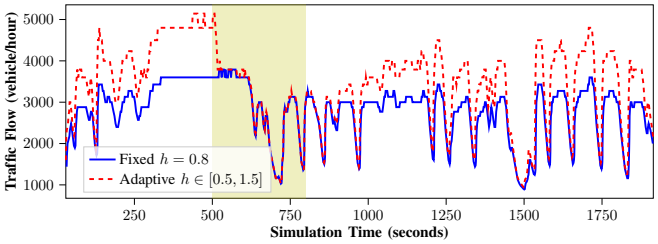
the minimum string-stable h was used. In Figures 7(c), we observe that during the loss process, Sample 3's traffic flow is slightly decreased when compared to the fixed headway scenario. However, the overall achieved traffic flow is 15.54% higher as seen in Table IV.

B. Network Outage

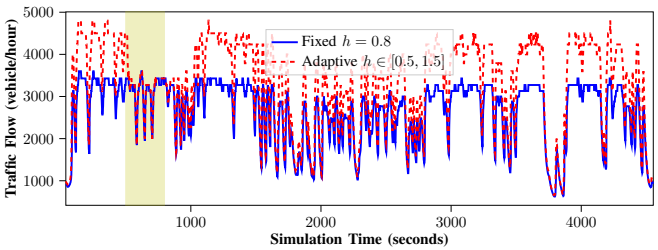
We tested our proposed adaptation on a scenario where total network outage occurred. Our early tests established that a minimum headway of $h = 1.5$ is string-stable under a total network outage scenario for the acceleration profiles we used. Therefore, in the case of a total network outage, the adaptation algorithm increases the headway to 1.5 during the loss process. Under this loss process, samples that can run string-stable at fixed $h < 1.5$ will have higher traffic flow during loss since total network loss always sets the headway to the maximum 1.5 as shown in Figure 8(d) and 8(e). In addition, recovering from a total network outage scenario is relatively slow over time and the adaptive algorithm gradually increases the traffic flow over time, where it matches the fixed algorithm at about

TABLE V
Traffic Throughput (*vehicle/hour*) Achieved Under a Network Outage Scenario With Different Acceleration Profiles.

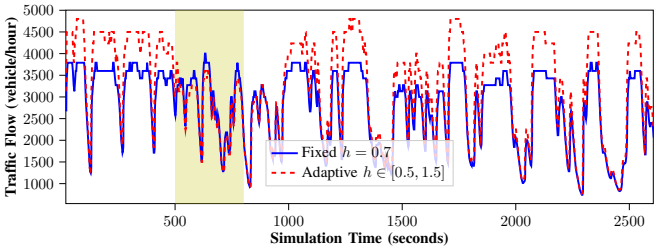
Acceleration Profile	Fixed h	Adaptive h	Improvement %
Sample 1	1813.95	2621.93	+44.54%
Sample 2	1761.46	2916.65	+65.58%
Sample 3	1737.8	2652.07	+52.61%
Sample 4	1680.66	2374.74	+41.3%
Sample 5	1953.8	2588.77	+32.5%



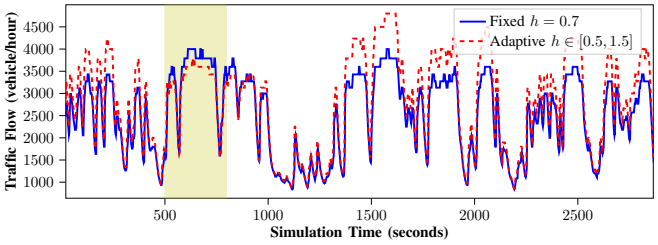
(a) Sample 1 flow comparison



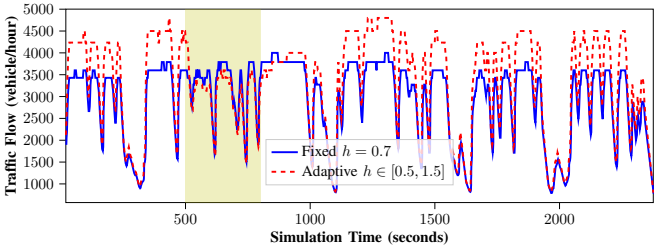
(b) Sample 2 flow comparison



(c) Sample 3 flow comparison

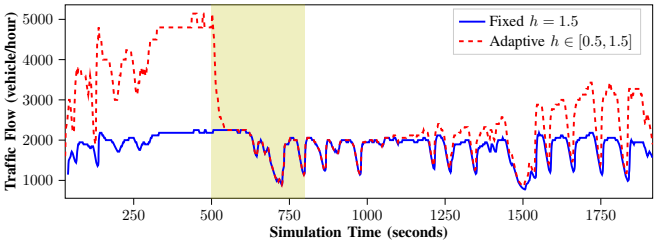


(d) Sample 4 flow comparison

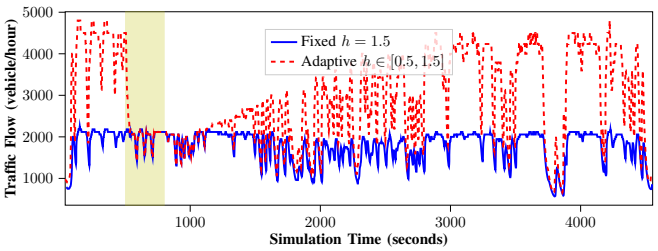


(e) Sample 5 flow comparison

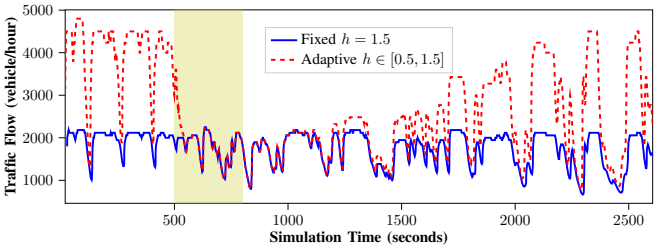
Fig. 7. Traffic Flow under 50/50 Packet Loss Scenario



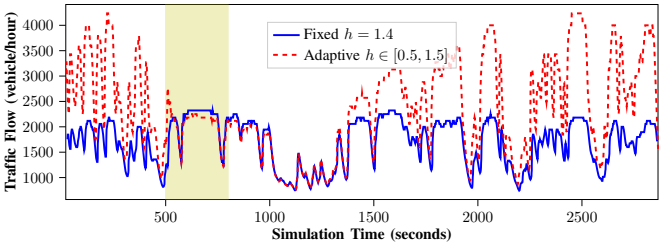
(a) Sample 1 flow comparison



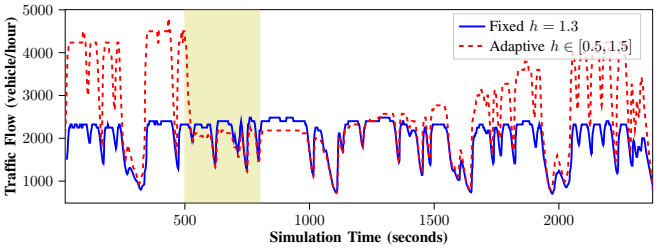
(b) Sample 2 flow comparison



(c) Sample 3 flow comparison



(d) Sample 4 flow comparison



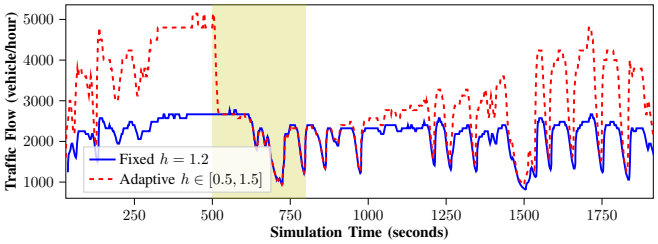
(e) Sample 5 flow comparison

Fig. 8. Traffic Flow under Total Network Outage Loss Scenario

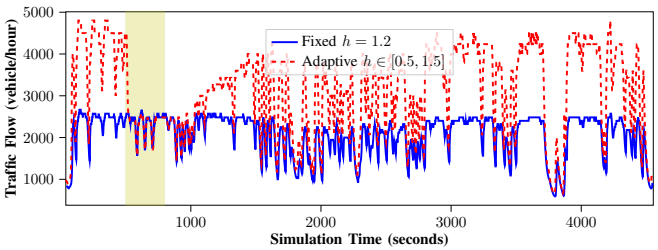
the 1100 seconds mark, and then increases the traffic flow afterwards.

C. Congested Network : $MGL = 10, MBL = 40$

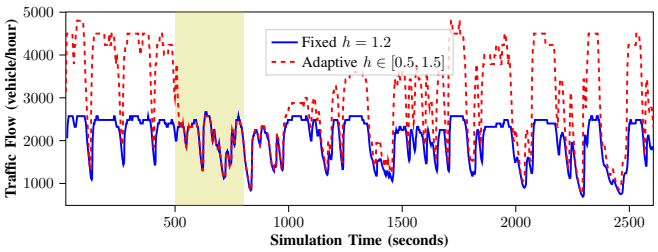
We applied our approach to a network with a loss process of $MGL = 10, MBL = 40$, and we noticed that our platoon adapted to the loss process well as seen in Figure 9. The remaining samples set h slightly higher than the minimum string-stable h, as our proposed solution is cautious when evaluation network quality. In the previous experiments with network outage, the adaptation algorithm assumes that the network is at its worse condition, and therefore, sets the h value to 1.5, its maximum value. We can apply that method to all loss processes on the expense of reduced traffic flow during



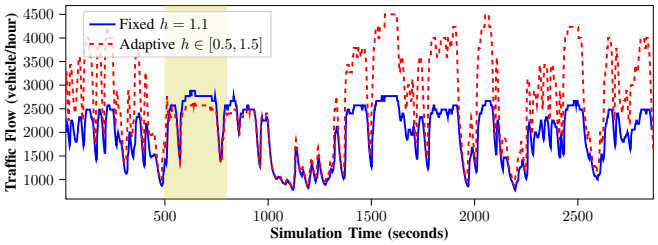
(a) Sample 1 flow comparison



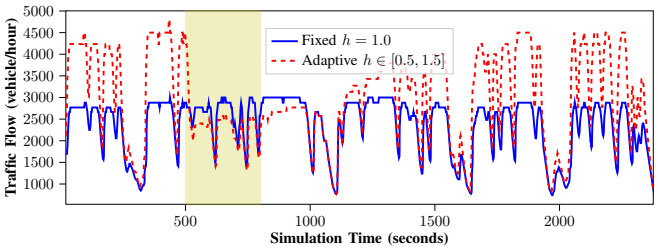
(b) Sample 2 flow comparison



(c) Sample 3 flow comparison



(d) Sample 4 flow comparison



(e) Sample 5 flow comparison

Fig. 9. Traffic Flow under Congestion $MGL = 10, MBL = 40$ Loss Scenario

loss. Therefore, the design of a platoon controller with adaptive h value may need to have a trade-off between being too cautious when increasing h value due to loss on the expense of reducing traffic flow during the loss event. It is also possible to incorporate factors such as degree of acceleration jerk and change to adapt the h value as well, but that is beyond the scope of this paper.

TABLE VI
TRAFFIC THROUGHPUT (*vehicle/hour*) ACHIEVED UNDER A NETWORK CONGESTION WITH DIFFERENT ACCELERATION PROFILES.

Acceleration Profile	**Fixed h**	**Adaptive h**	**Improvement %**
Sample 1	2121.77	2958.2	+39.42%
Sample 2	2046.14	3077.55	+50.41%
Sample 3	2018.25	2917.44	+44.55%
Sample 4	1946.37	2593.83	+33.27%
Sample 5	2318.36	2906.6	+25.37%

VI. CONCLUSION & FUTURE WORK

In this work, we demonstrated a method to adapt time headway of a one-vehicle look-ahead to the condition of the communication network in a way where it does not break the string stability of the platoon. When compared to using a fixed string-stable time headway, it achieves increased traffic throughput by an average of %34.56 for the loss processes we studied.

It is worth noting that the minimum string-stable headway value depends implicitly upon the acceleration and deceleration behavior of the platoon leader which is dictated by the acceleration profiles we used. It is also true that the minimum time headway increases as a function of platoon length. Although challenging to accomplish, development of an accurate model of these dependencies would be a valuable contribution to this problem domain.

Our work assumes a homogeneous platoon of ten vehicles. For heterogeneous platoons, it would be sufficient to derive the lower and upper bounds on the time headway from the maximum acceleration and deceleration capabilities of the least capable vehicles and to enforce those acceleration limits on the entire platoon. Nevertheless, the optimality of that approach is not obvious. Consequently, the impact of heterogeneity merits a thorough study.

ACKNOWLEDGMENT

Clemson University is acknowledged for generous allotment of compute time on Palmetto cluster.

REFERENCES

[1] "IEEE standard for information technology – Telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, Dec. 2016.

[2] L. Peppard, "String stability of relative-motion PID vehicle control systems," *IEEE Transactions on Automatic Control*, vol. 19, no. 5, pp. 579–581, 1974.

[3] D. Yanakiev, J. Eyre, and I. Kanellakopoulos, "Analysis, design, and evaluation of ACVS for heavy-duty vehicles with actuator delays," *PATH research report*, 1998.

[4] D. Swaroop and J. K. Hedrick, "Constant spacing strategies for platooning in automated highway systems," *Journal of Dynamic Systems Measurement and Control-transactions of The Asme*, vol. 121, no. 3, pp. 462–470, 1999.

[5] S. Knorn and R. H. Middleton, "Stability of two-dimensional linear systems with singularities on the stability boundary using LMIs," *IEEE Transactions on Automatic Control*, vol. 58, no. 10, pp. 2579–2590, 2013.

[6] J. Ploeg, B. T. Scheepers, E. Van Nunen, N. Van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 260–265.

[7] G. Naus, R. Vugts, J. Ploeg, R. van de Molengraft, and M. Steinbuch, "Cooperative adaptive cruise control, design and experiments," in *Proceedings of the 2010 American control conference*. IEEE, 2010, pp. 6145–6150.
[8] Z. Abdollahi, R. A. Biroon, and P. Pisu, "Cooperative adaptive cruise control: PDE modeling and stability analysis," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 5554–5559.
[9] J. Ploeg, E. Semsar-Kazerooni, G. Lijster, N. van de Wouw, and H. Nijmeijer, "Graceful degradation of cooperative adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 488–497, 2014.
[10] R. Vugts, "String-stable CACC design and experimental validation," *Diss. Technische Universiteit Eindhoven*, 2010.
[11] S. E. Shladover, "The California PATH program of IVHS research and its approach to vehicle-highway automation," in *Proceedings of the Intelligent Vehicles92 Symposium*. IEEE, 1992, pp. 347–352.
[12] X. Huppé, J. de Lafontaine, M. Beauregard, and F. Michaud, "Guidance and control of a platoon of vehicles adapted to changing environment conditions," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, vol. 4. IEEE, 2003, pp. 3091–3096.
[13] C.-Y. Liang and H. Peng, "String stability analysis of adaptive cruise controlled vehicles," *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, vol. 43, no. 3, pp. 671–677, 2000.
[14] C. Lei, E. Van Eenennaam, W. K. Wolterink, G. Karagiannis, G. Heijenk, and J. Ploeg, "Impact of packet loss on CACC string stability performance," in *2011 11th International Conference on ITS Telecommunications*. IEEE, 2011, pp. 381–386.
[15] Z. Wang, G. Wu, and M. J. Barth, "Developing a distributed consensus-based cooperative adaptive cruise control system for heterogeneous vehicles with predecessor following topology," *Journal of Advanced Transportation*, vol. 2017, 2017.
[16] P. Fernandes and U. Nunes, "Platooning with IVC-enabled autonomous vehicles: Strategies to mitigate communication delays, improve safety and traffic flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 91–106, 2012.
[17] A. Rayamajhi, Z. A. Biron, R. Merco, P. Pisu, J. M. Westall, and J. Martin, "The impact of dedicated short range communication on cooperative adaptive cruise control," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.
[18] A. N. Rayamajhi, "Exploring smart infrastructure concepts to improve the reliability and functionality of safety oriented connected vehicle applications," Ph.D. dissertation, Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 2019.
[19] C. Campolo, Y. Koucheryavy, A. Molinaro, and A. Vinel, "Characterizing broadcast packet losses in IEEE 802.11 p/WAVE vehicular networks," in *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE, 2011, pp. 735–739.
[20] P. Fazio, M. Tropea, C. Sottile, and A. Lupia, "Vehicular networking and channel modeling: a new Markovian approach," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2015, pp. 702–707.
[21] T. Blazek and C. F. Mecklenbräuker, "Low complexity SNR-based packet level burst-error model for vehicular ad-hoc networks," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.
[22] I. Bocharova, B. Kudryashov, M. Rabi, N. Lyamin, W. Dankers, E. Frick, and A. Vinel, "Characterizing packet losses in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8347–8358, 2019.
[23] M. E. Renda, G. Resta, P. Santi, F. Martelli, and A. Franchini, "IEEE 802.11p VANETs: Experimental evaluation of packet inter-reception time," *Computer Communications*, vol. 75, pp. 26–38, 2016.
[24] J. Ploeg, N. Van De Wouw, and H. Nijmeijer, "Lp string stability of cascaded systems: Application to vehicle platooning," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 2, pp. 786–793, 2013.
[25] NHTSA, "LIDAR speed-measuring device performance specifications," Washington, DC: National Highway Traffic Safety Administration, Tech. Rep. No. DOT HS 809 811, Mar. 2013.
[26] US Environmental Protection Agency, "EPA US06 or Supplemental Federal Test Procedure (SFTP)," https://www.epa.gov/emission-standards-reference-guide/epa-us06-or-supplemental-federal-test-procedure-sftp, accessed: 2020-03-11.
[27] "National Renewable Energy Laboratory (NREL), Transportation Secure Data Center - Drive Cycle Data," https://www.nrel.gov/transportation/secure-transportation-data/tsdc-drive-cycle-data.html, accessed: 2020-10-14.
[28] N. Lyamin, A. Vinel, M. Jonsson, and J. Loo, "Real-time detection of denial-of-service attacks in IEEE 802.11p vehicular networks," *IEEE Communications letters*, vol. 18, no. 1, pp. 110–113, 2013.

Adil Alsuhaim received his Bachelor's degree in Computer Science from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, and a Masters of Science from the University of Southern California in Los Angeles, CA, USA. From 2009 to 2013, he worked as a lecturer at the department of Information and Computer Science at KFUPM. He is currently a PhD candidate at the School of Computing in Clemson University in Clemson, SC. His research interest include intelligent systems, and currently focused on wireless mobile ad-hoc networks such as VANET applications and challenges.

Anjan Rayamajhi received his PhD degree in electrical and computer engineering from Clemson University in Clemson, SC. He worked as a Graduate Research Assistant and Technical Lead in the design and deployment of Clemson University Connected Vehicle Testbed. Responsible for installing and maintenance of the CVT, network performance evaluations and involvement in Connected Vehicle Safety and Mobility application development, demonstration and testing. He currently works as a V2X software engineer at Leidos.

James Westall received the MS degree in computer science and the PhD degree in mathematics from the University of North Carolina at Chapel Hill. He is a professor of computer science in the School of Computing at Clemson University. His research interests include measurement and modeling of computer systems and networks, computer graphics, and high-performance computing.

Jim Martin received the Ph.D. degree from North Carolina State University, Raleigh, NC, USA. He is an Associate Professor with the School of Computing, Clemson University, Clemson, USA. Prior to joining Clemson, he was a Consultant for Gartner, and prior to that, a Software Engineer for IBM. His current research projects include heterogeneous wireless systems and DOCSIS 3x cable access networks. He received funding from the US National Science Foundation (NSF), NASA, the Department of Justice, Cisco, IBM, CableLabs, and BMW. His research interests include broadband access, wireless networks, Internet protocols, and network performance analysis.