Quick answer to questions:

- The timestamps are not estimated.
  - There is a timestamp in the message sent from the cloud server, which is the time shown on server's clock when the server sends the message. The server synchronizes its clock every ten minutes to UTC time.
  - The vehicle also knows the time on its own clock every time when it sends and receives messages.

- Our goal is not to estimate the future timestamp, but instead how many seconds left until the end of current traffic light cycle.

- This PowerPoint include:
  - The detailed description of the problem.
  - Some modifications and additional explanations to 'slide 8'.

The problem we are trying to solve: predicting traffic light countdown

Problem description:

When our vehicle is approaching an intersection where traffic light is installed, we would like to know the amount of time or how many seconds left of current traffic phase – if current traffic light is green, knowing the countdown helps driver determined whether it is safe to pass and if current traffic light is red, it also helps to make the decision that whether it is worth shutdown the engine to reduce fuel consumption and meets environmental regulation requirements etc.

In our case, there is a cloud server, which is provided and operated by a third party, continuously collects data and predicts how many seconds remaining for (almost all) traffic lights located in particular cities.

With the help of V2I (Vehicle to Infrastructure) technology, our vehicle is able to send a BSM message to this cloud server at a frequency of ten times per second (10/s or 10 Hz). This cloud server receives all the messages and for every 300ms, this server checks these messages and decides whether if it is necessary to respond. If this server decides to respond to these messages, it will send a SPaT message to our vehicle that contains traffic light information including traffic light phase and countdown.

The problem we are trying to solve: predicting traffic light countdown

The nature of the problem:

However, the connection between our vehicle and this cloud server may not always be ideal at all locations, and therefore, some issues may occur, for example, our vehicle may not always receives the message in time but after a delay, the message our vehicle receives may not always be the most recently sent, or our vehicle may completely lose connection to the cloud server for a couple of minutes while the vehicle still expects critical data from the server.

A particular sub-problem:

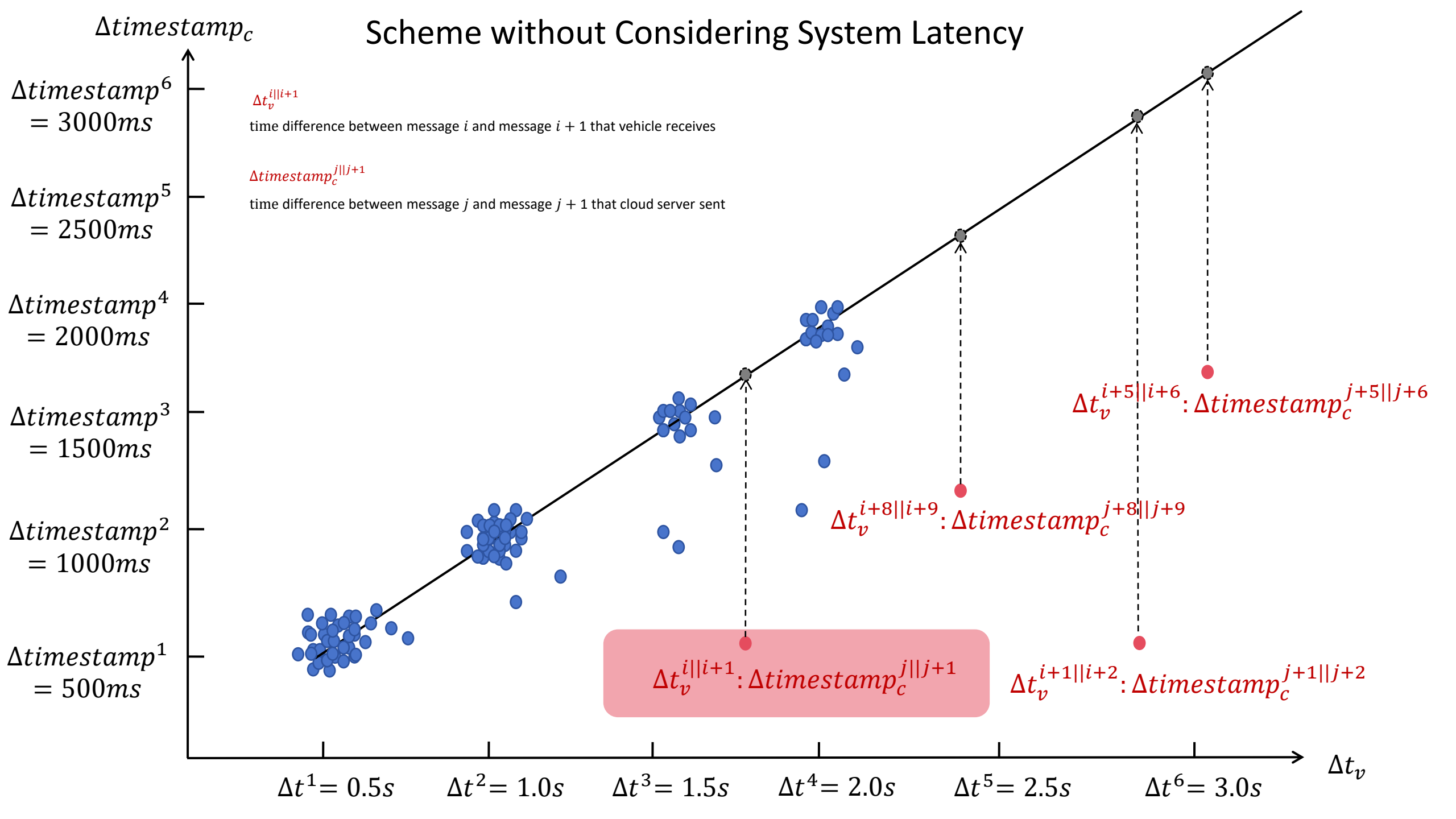
There is a delay in telecommunication between the cloud server and our vehicle caused by various factors, and therefore the message our vehicle receives is outdated to some degree that it is not accurately enough to provide to our customers. We would like to reduce this negative effect and applied an algorithm to adjust the data we received.

The problem we are trying to solve: predicting traffic light countdown

Assumption we made:

1. We assume the delay between the cloud server and our vehicle is *stable* during a certain period of time when vehicle is running in certain regions. For example, the latency for most of the message our vehicle receives *should be within two seconds* if this vehicle is running in San Francisco on Thursday under 40 miles per hour. And this *two seconds* is determined by the algorithm.

2. The clock running on the cloud server and the clock running on vehicle are independent and accurate.


Scheme without Considering System Latency
$\Delta timestamp_c$
$\Delta timestamp^6 = 3000ms$
$\Delta timestamp^5 = 2500ms$
$\Delta timestamp^4 = 2000ms$
$\Delta timestamp^3 = 1500ms$
$\Delta timestamp^2 = 1000ms$
$\Delta timestamp^1 = 500ms$
$\Delta t_v^{i||i+1}$
time difference between message i and message $i+1$ that vehicle receives
$\Delta timestamp_c^{j||j+1}$
time difference between message j and message $j+1$ that cloud server sent
$\Delta t_v^{i||i+1} : \Delta timestamp_c^{j||j+1}$
$\Delta t_v^{i+1||i+2} : \Delta timestamp_c^{j+1||j+2}$
$\Delta t_v^{i+8||i+9} : \Delta timestamp_c^{j+8||j+9}$
$\Delta t_v^{i+5||i+6} : \Delta timestamp_c^{j+5||j+6}$
Δt_v
$\Delta t^1 = 0.5s$
$\Delta t^2 = 1.0s$
$\Delta t^3 = 1.5s$
$\Delta t^4 = 2.0s$
$\Delta t^5 = 2.5s$
$\Delta t^6 = 3.0s$

explanation to slide 8:

We would like to train a model that helps decide the latency, by comparing With the help of V2I technology, we are able to establish a bi-directional telecommunication between the cloud server and the vehicle using UDP protocol, and in the SPaT message that the cloud server sends to the vehicle, there is a ***timestamp*** which indicates the exact moment when the cloud server sends the message.

In *slide* 8, the vertical axis shows the time difference between the time stamps coined in two consecutive messages the cloud server sends, hence the subscript c, and the vertical axis is named $\Delta timestamp_c$. Notation $\Delta timestamp_c^{j||j+1}$ therefore represents the length of the time interval between two messages sent by the cloud server. The superscript j and j+1 can be understanded as j-th and j$+1$ -th message.

Similarly, the horizontal axis shows the time difference between the two times when the vehicle receives SPaT message and is named Δt_v.

So, when we look at a red dot, it shows the time difference between two messages sent from the cloud server by looking at the vertical axis and it also shows the time difference between two messages received by the vehicle by looking at horizontal axis.

In effect, this graph is comparing the time intervals (so we see lots of Δt), and assuming when connection is good, these intervals should not vary much, so we see clusters of blue dots, which are *normal* dots, and the red dots are considered *abnormal,* and when a red dot appears, it is very likely that there is a delay and thus the information in the message should be adjusted.

The problem we are trying to solve: predicting traffic light countdown

Additional explanation to slide 8:

We would like to train an algorithm that helps decide the latency. The blue dots are training data and the line can be think of the *trained algorithm* that learns the latency. Red dots are messages the vehicle actually receives when running on the roads, the dashed arrow shows that these messages are adjusted by the algorithm.

For example, if *message j* sent from server indicates that currently traffic light is green and there is 33 seconds left, and vehicle receives the same message in time and label it *message i*. Half of a second later the server sends another message, *message j+1,* indicating that currently traffic light is green and there is 32.5 seconds left. However, when the vehicle receives this *message j+1*, and label it message *i+1*, model checks and sees 1.7 seconds have passed between i-th and i+1 -th message, so the model should determine there is likely a delay and information "there is 32.5 seconds left for green light" is outdated, it is more likely that there is 31.3 seconds left, because there is likely another 1.7 - 0.5 = 1.2 seconds have passed since the server sent the message.

This is a simplified scenario, and exact adjustment may vary depending on the model, but the idea remain the same.