Red Alliance GameData Issue at NYRO in match QF1

Rob Heslin

About Author:

I have been a Robot Inspector for 15 years and a Lead Robot Inspector for 8 years. I have been involved in Beta testing for many years, including this past year where I did test GameSpecificData. For the past 5 years I have trained as a CSA. I am also a current mentor on 340 and 3015.

Incident Description:

At NYRO, in QF1, the first seeded alliance(the red alliance) made up of FRC teams 340, 1507 and 3550, functioned in a way that looked like their alliance received the wrong GameData about the switch/scale configuration. The GameData should have been LLL for the red alliance. 340's robot drove to the wrong scale plate, the one on their right. 340 claims their code would not move if there was no GameData. 1507's robot did not drive forward at all, which according to the team would happen if they received no GameData. Their robot, according to them, should have scored on the switch. 3550's robot drove forward to cross the auto line, but they say GameData would not affect them as they were not using it. The alliance lost the match, so they were concerned.

Match Video Here: https://www.youtube.com/watch?v=080XP99KlgI

How I Became Involved:

I did not see QF1. So after the match 340, a team on which I am a mentor, asked me to look at what happened during the autonomous mode. They claimed that they ran the wrong auto based on the GameData they received. I spoke with Liz Smith(the FTA) and, given my background as a CSA and a 2018 beta tester, began investigating by looking at the logs of each team on the impacted alliance.

Logs, Data and Code:

Looking at the logs from each team, their DriverStation's each receive the GameData, reading out the line "Game Specific DataLLL" before the robot is enabled.

340:

1:37:47.929 PM	Game Specific DataLLL Boolean condition: true SENSOR 1: false SENSOR 2: false Boolean condition: true
1:37:48.065 PM	Boolean condition: true SENSOR 1: false SENSOR 2: false Boolean condition: true SENSOR 1: false SENS
1:37:48.163 PM	SENSOR 1: false SENSOR 2: false Boolean condition: true SENSOR 1: false SENSOR 2: false Boolean co

1507:

4:37:04.831 PM	Game Specific DataLLL
4:37:08.102 PM	*****Invalid Game Data [0]**** CenterStart AutoGameData: AutonomousInit
4:37:10.959 PM	Info roboRIO: Disk Free 228 MB, Memory Free 112 MB. DS Laptop: CPU 26% Batt 95%.

3550:

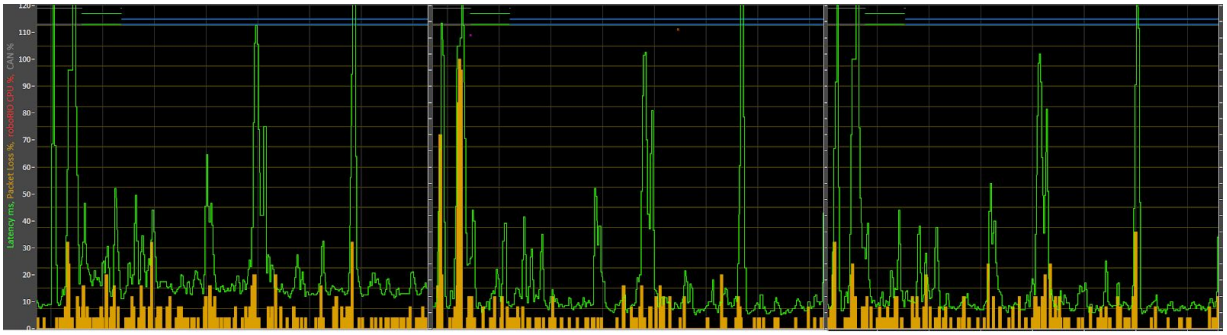
1:37:51.353 PM	Game Specific DataLLL
1:37:53.524 PM	Info roboRIO: Disk Free 176 MB, Memory Free 88 MB. DS Laptop: CPU 20% Batt 18%.
1:37:54.593 PM	AutonomousPeriodic

Here is basic info on all three teams' language,image versions, etc.

Team #	340	1507	3550
Language	Java 2018.4.1	C++ 2018.1.1	Java 2018.4.1
RoboRIO Image	2018_v17	2018_v16	2018_v17
DriverStation	18.0.1	18.0	18.0.1

FMS reported abnormal trip times and packet loss for every team on the alliance. Here are the graphs:

340: 1507: 3550:



Each shows similar patterns of increased trip time(green line) and packet loss(orange bars), with seemingly identical trip time peaks about 2 seconds before autonomous began when one might expect GameData to be sent. Reviewing all matches for each of the three teams during the day, no pattern like this, or trip time and packet loss spikes like this are seen. Graphs from the opposing alliance were not collected, so I can not check to see if similar spikes are seen in their logs.

Here is a snippet of 340's robot code that pulls GameData:

```
public static String FMSData() {
        return DriverStation.getInstance().getGameSpecificMessage();
}

public void autonomousInit() {
        String fms = FMSData();
        start = getStart();
        switch(fms) {
```

This is inline with FIRST's direction on when to pull the GameData per the screensteps directions:
http://wpilib.screenstepslive.com/s/currentCS/m/getting_started/l/826278-2018-game-data-details

Team 340's Repo: https://github.com/Greater-Rochester-Robotics/PowerUp2018-340/

Other teams' code was not acquired or reviewed.

On The Blue Alliance:

Looking at previous matches of the blue alliance in the QF1 match. Some had repeat GameData others did not. I did not collect logs from the alliance. I cannot say if they experienced a similar problem, or were affected by the same issue.

Analysis:

It is clear that the GameData was received by each DriverStation about 3 seconds before the robots began running autonomous mode. But from 1507's log their robot seems not to have received GameData, per the line "*****Invalid Game Data [0]****". The form and timing of this print statement implies they get the code once during AutonomousInit(). And 340 claim they ran code that would be for an R scale when they had an L scale. The difference in DriverStation program likely leads to the root of the different behavior. 1507's 18.0 DriverStation has a blank GameData after the program restarts, where 340's 18.0.1 DriverStation would retain the GameData from the previous match, which did have an R scale. That match is here:

https://www.thebluealliance.com/match/2018nyro_qm82

Having difficulty viewing 340's log, prompted further investigation. They were using one print statement ~80 time a second. Their code is also throwing a no USB camera detected fault as well. And there are numerous other print statements once their Autonomous and TeleOp after start. Additionally they are pushing at least 3 doubles in disabledPeriodic() over the SmartDashboard network table, which are likely different values each code cycle. I did not check the other teams' code at this time, as I am a mentor on 340, I have easy access to their code, but would not be surprised if similar issues might be in other code. This could have been the source of the trip times, but this is not supported by looking at all other matches, as none that day show similar patterning.

I do not have access to FMS logs for any of this, so I can not review for outside sources to the abnormal trip times. WiFi related problems would appear on the other alliance's logs, but those were not collected, so there is no way for me to compare them. It is suspicious that all three logs show peaks in the same relative pattern.

Theory:

The abnormal trip time/packet loss slowed the receipt of the GameData packet. It is a TCP packet, but in beta testing we found it could take up to 3-4 seconds to get the the robot, so it can be delayed for several reasons. Because it is TCP, it would have gone through eventually. But could be too late for teams to have used it , even if the teams are using the GameData retrieval form FIRST recommends i.e. at the very beginning of autonomous. Though I provide no experimental evidence that trips times and packet loss are what delayed GameData.

Next Match:

In QF5, the next match between the two alliances in QF1, the only code change I am aware of on the red alliance was the removal of the print statement that was printing a lot to the DriverStation. The trip time was lower, as was the packet loss, but there is no proof that the code change caused this. According to the red alliance teams, their robots responded to the GameData as they should.

Ancillary Note:

In QF5, 3986, a team on the blue alliance, scored in the red alliance scale during autonomous. Fearing issues in line with QF1, I reviewed the log files for that match at the request of the FTA. The logs showed the DriverStation received the GameData “LLL”, and the following line was a print statement from their code that ended with “LLL_droite”.(the team is from Quebec) This makes sense as their robot was on the right side of the field and their scale was on the left. So we believe the issue was that their own code went the wrong way, and their left side scale code was for the right scale.

Things Teams Can Improve:

-Print the GameData back when autonomous choice is made, this can help with explaining why a robot reacts a certain way. Use a print statement so it appears in the logs.

-Have drivers or technicians delete GameData before connecting to the robot, then make sure that the robot does not begin autonomous until a valid GameData string is seen

-If autonomous is short, try waiting a few seconds to run autonomous, and check then for GameData

-Check GameData halfway through autonomous to confirm the first choice is right, perhaps before scoring so teams do not score against themselves.

-Reduce print statements and SmartDashboard data pushes and keep code clear of errors or other extra data (there is no proof that is what happened here, but it would not hurt the situation)