Introduction To Multicasting

Robert Hirsh

NASA/JSC ER6

Software Robotics and Simulation Division

Mar 13th 2019

Robert.L.Hirsh@nasa.gov

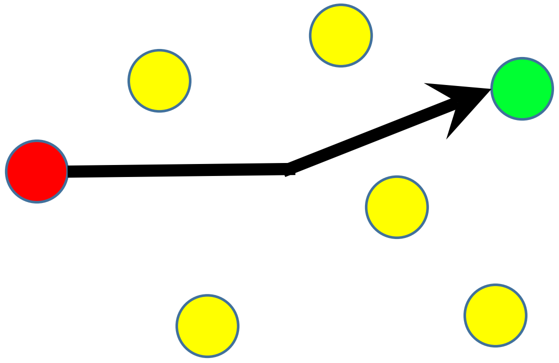
Outline

- Background on Multicast
- AA2 Components
- Data Player App (test.jar)
    - Telemetry Player
    - Telemetry Receiver
    - Telemetry Format
        - Units/meaning of data items
- Example Packet Dissection
- Questions?

Multicast

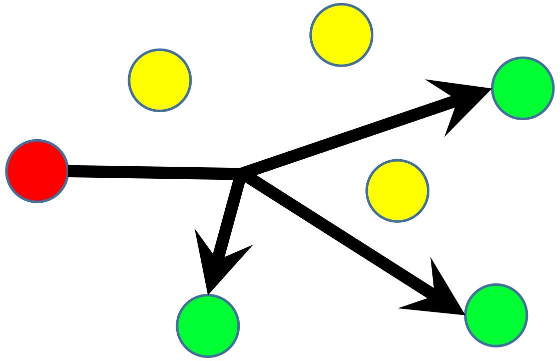
- Computers communicate by sending Ethernet messages
- Unicast message are point-to-point
  - One sender and one receiver
  - Like a text or phone call
  - Each computer has a unique IP address (4 numbers)
    - www.google.com is 172.217.15.68
    - www.apple.com is 17.142.160.59
- Multicast is like a radio or TV broadcast.
  - Sender sends out 1 message regardless if one, ten, 1000, or a million clients are receiving
  - Removes overhead of sending individual copy to each one
  - No rebroadcast if a message is missed by one client

Unicast



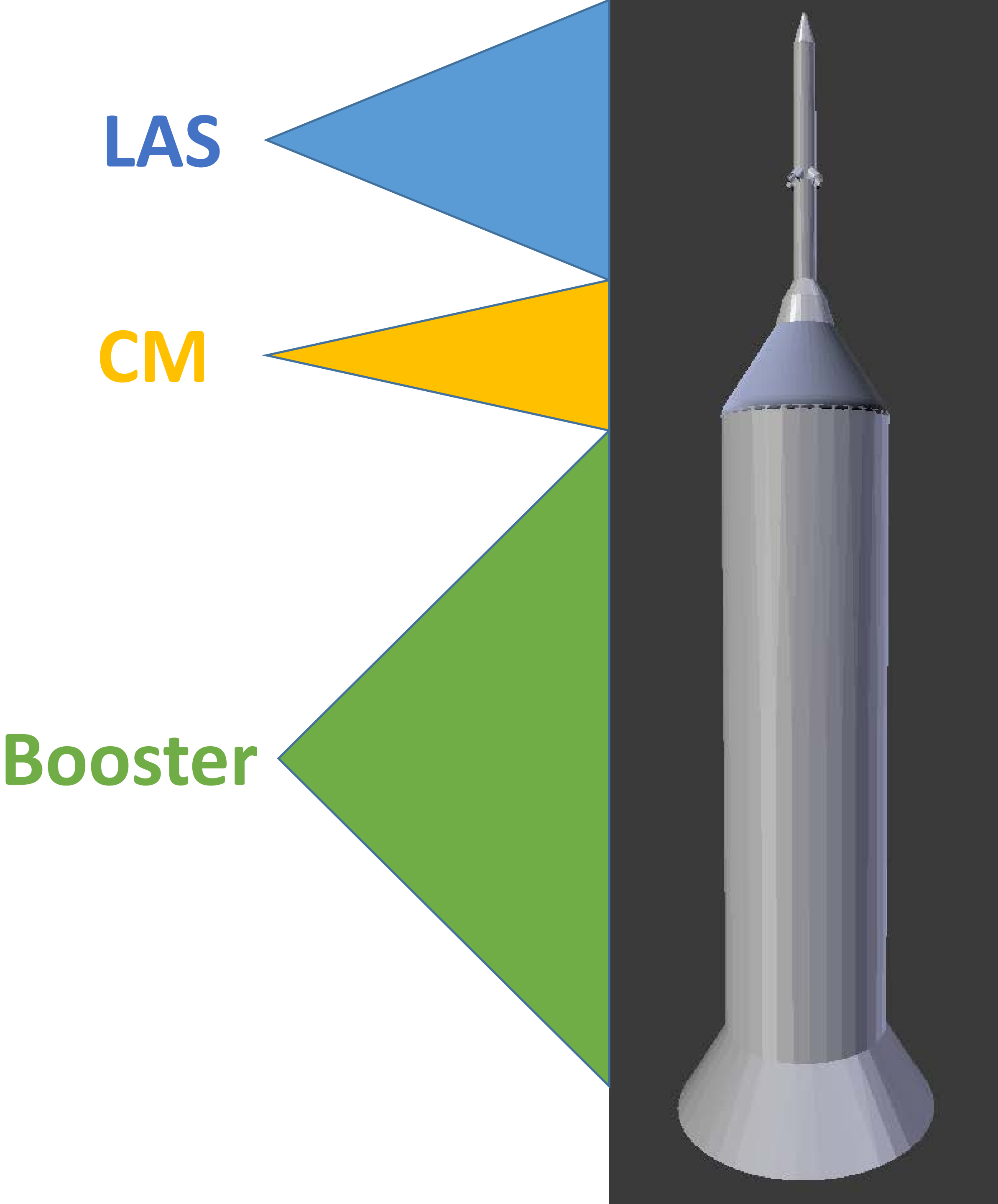
Red is source, Green is destination
Yellow are other computers.
See more at:
https://en.wikipedia.org/wiki/Multicast

Multicast



Vehicle Components

- There are 3 “pieces” of AA2
  - The Launch Abort System (LAS)
  - The Command Module (CM)
  - The Launching Rocket (Booster)
- All 3 Pieces of launch together
- After reaching the test altitude/speed the abort will occur
  - LAS/CM pull away from the Booster
- After LAS/CM are safely away, the LAS jettisons from the CM
- Individual Wavefront (.obj) models for these 3 pieces have been provided



Telemetry Player

- NASA is providing a telemetry player
  - Sends out a Multicast UDP version of the data in the Data.csv file
  - Each row of file is sent as a separate Ethernet packet
  - Your app needs to read this multicast data
  - Live vehicle data will be sent in same format
- Exact method of receiving multicast data varies based on software you use to design your app.
  - There are many examples of Multicast clients available online
  - Sample Java Multicast Receiver is included in test.jar file
    - Prints out the content of the message, does not do anything with the data
    - Feel free to use the code as a starting point if you are using Java for your application, but don't feel like you need to

Telemetry Player instructions

- Once you have downloaded the test.jar and Data.csv files, place them in a folder
- Open a command prompt (in that folder)
- Type the following command:

```
java -jar test.jar
```

- That will start the data player and you will see print messages as it sends out each line of the Data.csv file
- You can edit the data file to send other sample data, as you wish
  - Feel free to tweak data trajectory for your final video
- By default, it will send data on Multicast address (237.7.7.7), on port (42055), at normal (1x) rate, from the Dats.csv file.
  - Those parameters can all be changed if needed by using additional command line arguments.
  - For example: to replay the file at 3x normal speed, type the following:

```
java -jar test.jar -s 3
```

- To see all command line arguments type

```
java -jar test.jar -h
```

Telemetry Reader

- Open a command prompt (in the folder with test.jar)
- Type the following command:

```
java -cp test.jar receiver
```

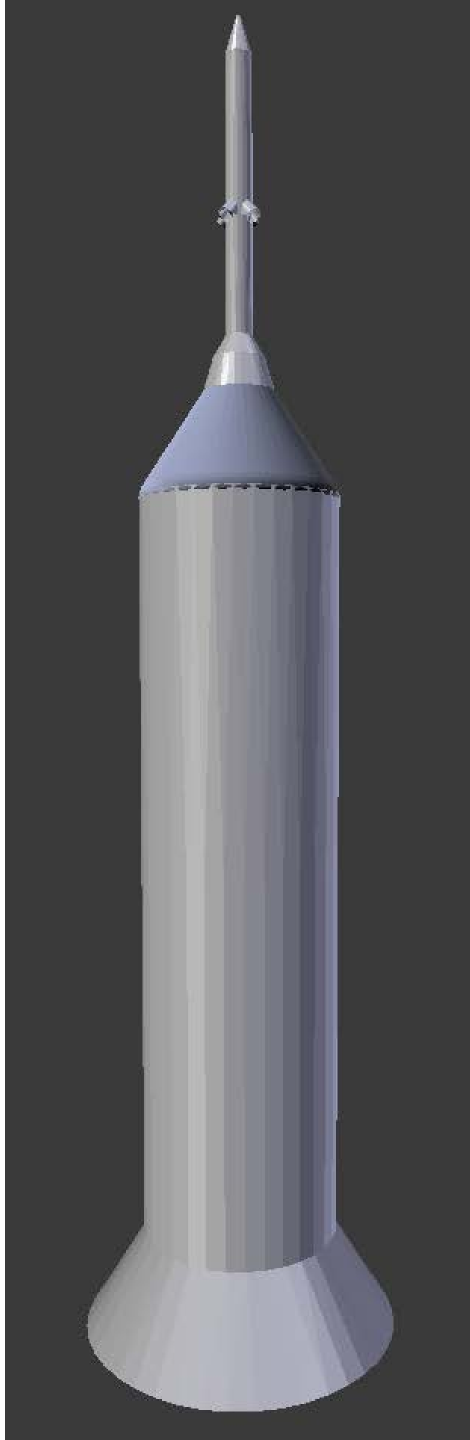
- That will start the data receiver, and if a telemetry player is running, the receiver app will print out a text version of all the parameters that it receives
  - Note: You can open up multiple instances of the receiver (anywhere in the same local network) and each will receive the data
- To see the source code in the tar file, type the following to extract it:

```
jar -xvf test.jar
```

  - Feel free to modify/examine any of the java files insider.
- Note: The player listens to Multicast address (237.7.7.7) on port (42055)
  - Those parameters have no command line arguments
    - Feel free to add that feature to receiver.java if you want to (see sender.java “GetOpts” example)

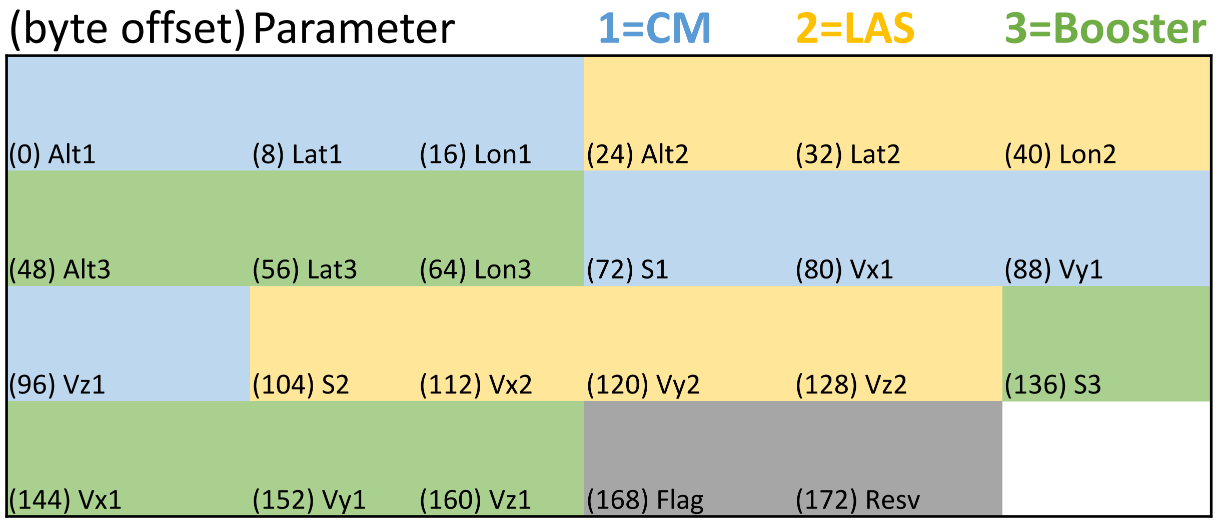
Telemetry Format

- Data stream has location/orientation for 3 AA2 pieces
- 7 parameters in the telemetry (per piece)
  - 3 for position, and 4 for orientation
  - All of these items are 8-byte floating point numbers (i.e. a double)
  - 3 Positions are: Elevation (meters), and Latitude/Longitude (radians).
  - Orientation is a quaternion: 4 unitless numbers in range between -1 to 1
    - Separate talk concerning quaternions will in Live Connect 3
- In addition to those 21 (3 object, 7 per object) doubles, there are two 4-byte integers at the end of each message
  - One is a flag for when the Booster and LAS are “active”
    - Use for any smoke/fire animations in your application
    - 0 means no engines active, 1 is booster only, 2 is LAS only, and 3 is both.
  - Second parameter is reserved for future use, but may be used in future
    - Let us know ideas for what you think would be useful to put into that slot



Telemetry Payload

- 176 bytes of data in each message
  - Items are packed same order as Data.csv
  - Shown graphically above
  - Doubles takes up 8 bytes. The last 2 items (in grey) are 4-bytes



(byte offset) Parameter 1=CM 2=LAS 3=Booster

(0) Alt1	(8) Lat1	(16) Lon1	(24) Alt2	(32) Lat2	(40) Lon2
(48) Alt3	(56) Lat3	(64) Lon3	(72) S1	(80) Vx1	(88) Vy1
(96) Vz1	(104) S2	(112) Vx2	(120) Vy2	(128) Vz2	(136) S3
(144) Vx1	(152) Vy1	(160) Vz1	(168) Flag	(172) Resv	

- Raw byte dump of first rows of data file is shown on next slides
  - For reference on converting 8 raw bytes to doubles, you can use
  - https://gregstoll.com/~gregstoll/floattohex
    - Make sure to tic “swap endinness” box

Packet1 (1st row)

CM Altitude = -47.08564 CM Latitude = 0.49669439

```
5e 9d 63 40 f6 8a 47 c0  df 06 4a 44 d7 c9 df 3f
11 c6 4f e3 de 7c f6 bf  5e 9d 63 40 f6 8a 47 c0
df 06 4a 44 d7 c9 df 3f  11 c6 4f e3 de 7c f6 bf
5e 9d 63 40 f6 8a 47 c0  df 06 4a 44 d7 c9 df 3f
11 c6 4f e3 de 7c f6 bf  8a c8 37 32 4b 06 dd 3f
b3 0e c3 1e 45 63 e1 3f  bc 35 55 9c 0f f4 dc 3f
db d7 e4 9d 01 5d e1 bf  8a c8 37 32 4b 06 dd 3f
b3 0e c3 1e 45 63 e1 3f  bc 35 55 9c 0f f4 dc 3f
db d7 e4 9d 01 5d e1 bf  8a c8 37 32 4b 06 dd 3f
b3 0e c3 1e 45 63 e1 3f  bc 35 55 9c 0f f4 dc 3f
db d7 e4 9d 01 5d e1 bf  00 00 00 00 ff ff ff ff
```

Booster Quat Z = -0.54260331 Flags = 0 reserved = -1

Packet2 (2nd row)

CM Altitude = -47.086411

CM Latitude = 0.49669439

```
e0 81 01 84 0f 8b 47 c0 df 06 4a 44 d7 c9 df 3f
11 c6 4f e3 de 7c f6 bf e0 81 01 84 0f 8b 47 c0
df 06 4a 44 d7 c9 df 3f 11 c6 4f e3 de 7c f6 bf
e0 81 01 84 0f 8b 47 c0 df 06 4a 44 d7 c9 df 3f
11 c6 4f e3 de 7c f6 bf 66 cd db 96 19 06 dd 3f
36 19 d9 be 15 63 e1 3f 06 df b0 44 d2 f4 dc 3f
a1 b8 a9 a2 f4 5c e1 bf 66 cd db 96 19 06 dd 3f
36 19 d9 be 15 63 e1 3f 06 df b0 44 d2 f4 dc 3f
a1 b8 a9 a2 f4 5c e1 bf 66 cd db 96 19 06 dd 3f
36 19 d9 be 15 63 e1 3f 06 df b0 44 d2 f4 dc 3f
a1 b8 a9 a2 f4 5c e1 bf 00 00 00 00 fe ff ff ff
```

Booster Quat Z = - 0.54259712

Flags = 0

reserved = -2