Specification of UDP communication

20151013; Petter Wirfält; petter.wirfalt@consultant.volvo.com

Overview

Two independent systems are continuously transmitting UDP packets from the real-time platform ("autobox", AB). One system transmits (raw) data from the forward mounted LIDAR system, and one system transmits ego-, lane-, and "track" (i.e., the states and other parameters of objects that are being tracked) -output from the Sensor Data Fusion (SDF) system also running on the AB. At the very beginning of every UDP package sent there is an identifier, which allows a listener to determine from which transmission system the package originated. Hence a listener can opt to only decode the messages relevant to its particular application.

Transmission details

The UDP packets will originate from IP 169.254.145.80. It is desirable that the receiver is on the same subnet (with mask 255.255.0.0). The packets will be transmitted as a broadcast to IP 255.255.255.255, on port 2001 for the LIDAR data, and port 13000 for the SDF data.

Due to hardware/software limitations of the AB, the maximum size of a UDP packet is 1514byte (including UDP/TCP/IP headers, etc); of these bytes, there is space for 1472 bytes of data. The messages to be transmitted contain more data than 1472 bytes. Hence, a message is partitioned into several packets, and prior to that a message can be decoded, a number of packets must be merged into a message.

4 bytes of the data are used for packet identification. These bytes are used for separating LIDAR data from SDF data, and also for specifying the current packet's position in the message.

LIDAR packets:

Packets are numbered 0 to 11. The packet number is added to a "magic word" key which is a uint32 with hex value 'FEDCBA98'. Thus each packet will contain an identifier, ranging from 'FEDCBA98' to 'FEDCBAA3'. Hence the packets can be structured to a message. The packets are sent with a delay of about 5ms. After the 12^{th} packet, a new message beginning with packet number 0 starts after about 25ms. This gives a message throughput of 12.5Hz, which is identical to the IBEO scan/data-output frequency. (Please observe little or big endian reception; the order of the identifying magic word bytes might thus be reversed)

SDF packets:

Packets are numbered 0 to 3. The packet number is added to a “magic word” key which is a uint32 with hex value ‘F0E1D2C3’. Thus each packet will contain an identifier, ranging from ‘F0E1D2C3’ to ‘F0E1D2C6’. The packets are sent with a delay of about 5ms. After the 4th packet, a new message beginning with packet number 0 starts after about 10ms. This gives a message throughput of 40Hz, which is identical to the SDF-output frequency. (Please observe little or big endian reception; the order of the identifying magic word bytes might thus be reversed)

Message details

Please follow the below chart for decoding the messages. All data is transmitted as uint32 words, and it is assumed that it is received as such.

1) The first word in each packet is the magic word. Once packet identification is completed, and a complete message has been formed, this word is to be discarded.
2) Stack the remaining words of the current message packets into a vector. For the LIDAR data, this vector should have 4404(=12x367) elements, and for the SDF data, the vector should have 1468 (=4x367) elements. In both cases, the first word of the first packet of the message should be the first element of the vector.
3) Decode the word-vector into a uint8 vector:
    a. LIDAR: use little endian byte order. The resultant vector has 17616-element.
    b. SDF: use big endian byte order. The resultant vector has 5872 elements.
4) Referencing to the attached message specification, the message contents are decoded from the received uint8 bytes.
    a. For both message types, some elements contain data that does not change from message to message. Other elements (e.g., timestamps) change in a predictable manner. By examining the decoded contents of such elements, it is possible to verify the proper functioning of the decoder.

LIDAR UDP message specification

The LIDAR UDP message is encoded in the order and with the data-types as specified in Table 1.

Table 1: LIDAR scan data specification

NAME	TYPE	BYTE/ ELEMEN T	# OF ELEMEN TS	ELEMENT PAY-LOAD (BYTES)	STARTING OFF-SET (BYTE)	ENDING OFF-SET (BYTE)	COMMEN T
timeMessDataCrea te	doubl e	8	1	8	1	8	AB clock
sizeOfThisMess	uint3 2	4	1	4	9	12	

scanNo	uint16	2	1	2	13	14	
scanStatus	single	4	1	4	15	18	
syncPhaseOffset	double	8	1	8	19	26	
scanStartTimeNTP	double	8	1	8	27	34	
scanEndTimeNTP	double	8	1	8	35	42	
angTicksPerRotation	uint16	2	1	2	43	44	
startAng	single	4	1	4	45	48	(in deg)
endAng	single	4	1	4	49	52	(in deg)
scanPts	uint16	2	1	2	53	54	
mountingPosYawAng	single	4	1	4	55	58	(in deg)
mountingPosPitchAng	single	4	1	4	59	62	(in deg)
mountingPosRollAng	single	4	1	4	63	66	(in deg)
mountingPosX	single	4	1	4	67	70	(in m)
mountingPosY	single	4	1	4	71	74	(in m)
mountingPosZ	single	4	1	4	75	78	(in m)
flags	uint16	2	1	2	79	80	
scanPtLayer	uint8	1	1000	1000	81	1080	
scanPtEcho	uint8	1	1000	1000	1081	2080	
scanPtFlags	uint8	1	1000	1000	2081	3080	
scanPtHorizontalAng	single	4	1000	4000	3081	7080	(in deg)
scanPtRadialDist	single	4	1000	4000	7081	11080	(in m)
scanPtEchoPulseWidth	single	4	1000	4000	11081	15080	(in m)
scanPtRsv	uint16	2	1000	2000	15081	17080	

For information (in addition to that given in Table 1) regarding the different elements, see below pdf, starting from page 7. NOTE that Table 1 has precedence – i.e., if a certain element is specified in two different formats, Table 1 is giving the correct format.

[ibeo LUX Familiy Ethernet Manual v1.36.pdf]

SDF UDP message specification

The UDP messages sent contain the fields in Table 2. The choice of this particular structure is due to a previously existing Volvo project. Note that some fields do not contain any data (i.e., some are 0). In addition, the length of the message is fixed; hence many of the last objects will not exist and thus all the data for these will be 0.

Table 2: UDP message byte specification

Signal position num/byte	Signal name	Siz e	Signal type	Signal description

0 / 0	sequenceNumber	32	int32	UDP message sequence number
1 / 4	timestamp	64	double	SW timestamp
2 / 12	interfaceVersion	32	int32	Interface version (here decimal 5002) Note: The format until here, first 128 bits, must be the same in all future versions.
3 / 16	numObjects	8	int8	Number of objects sent, valid or invalid (96)
4 / 17	numTrails	8	int8	Number of trails sent, valid or invalid (0)
5 / 18	coordSystem	8	int8	Coordinate system 0=Relative to ego, 1=Global
6 / 19	reserved	32	int32	Internal usage by TCJ viewer. Shall be set to zero in files/UDP.
7 / 23	egoVehType	8	int8 (enum)	0 = UNDETERMINED, 1=CAR, 2=MOTORCYCLE, 3=TRUCK, 4=PEDESTRIAN, 5=POLE, 6=TREE, 7=ANIMAL, 8=GOD, 9=BICYCLE, 10=UNIDENTIFIED_VEHICLE
8 / 24	egoWidth	32	single	Ego vehicle width in meters
9 / 28	egoLength	32	single	Ego vehicle length in meters
10 / 32	egoHeight	32	single	Ego height in meters
11 / 36	egoCSOffset	32	single	Ego coordinate system offset, longitudinally, from front of vehicle (relative coordinates)
12 / 40	egoSpeed	32	single	Ego vehicle velocity, m/s
13 / 44	egoAcc	32	single	Ego vehicle acceleration, m/s^2
14 / 48	egoLongPos	32	single	Longitudinal position of ego, meters (global coordinates)
15 / 52	egoLatPos	32	single	Lateral position of ego, meters (global coordinates)
16 / 56	egoHeadingAngle	32	single	Object heading angle, rad (global coordinates)
17 / 60	egoYawRate	32	single	Ego vehicle yaw rate
18 / 64	egoLatitude	32	single	Ego vehicle GPS latitude, degrees
19 / 68	egoLongitude	32	single	Ego vehicle GPS longitude, degrees
20 / 72	laneValid	8	int8	0 = invalid, > 0 = valid Same as for objValid (special for fake targets)
21 / 73	laneLength	32	single	Length of the visible lane
22 / 77	laneWidth	32	single	Lane width in meters
23 / 81	laneCurvature	32	single	Lane curvature
24 / 85	laneCurvatureRate	32	single	Lane curvature rate
25 / 89	laneLatOffset	32	single	Lateral offset of lane, left edge
26 / 93	laneHeadingAngle	32	single	Lane heading angle, rad
Objects begin here; 'numObjects' objects placed after each other				
27 / 97	objValid	8	int8	0 = invalid, > 0 = valid (for fake targets internal UDP format: 0 = invalid, 1=keep as they are, don't use the positions etc, 2=new/updated target, use positions etc)
28 / 98	objId	32	int32	Object ID
29 / 102	objVehType	8	int8 (enum)	0 = UNDETERMINED, 1=CAR, 2=MOTORCYCLE, 3=TRUCK, 4=PEDESTRIAN, 5=POLE, 6=TREE, 7=ANIMAL, 8=GOD, 9=BICYCLE, 10=UNIDENTIFIED_VEHICLE, 11=PIANO, 23-37= SPEED_LIMIT_SIGN_20 - SPEED_LIMIT_SIGN_160, 52=NO_ENTRY_SIGN, 53=NO_MOTORIZED_VEHICLES_SIGN, 55=NO_OVERTAKING_SIGN, 57=STOP_SIGN, 60=CURVE_WARNING_SIGN, 61=ROAD_WORK_WARNING_SIGN, 64=LEVEL_CROSSING_WITHOUT_GATE_SIGN and some more signs

30 / 103	objTrackingModel	8	int8 (enum)	0 = Cartesian, 1 = bicycle model, 2 = polar
31 / 104	objLongPos	32	single	Longitudinal position of object, meters
32 / 108	objLatPos	32	single	Lateral position of object, meters
33 / 112	objHeadingAngle	32	single	Object heading angle, rad, positive counterclockwise
34 / 116	objSpeed	32	single	Object speed, m/s (bicycle model)
35 / 120	objAcceleration	32	single	Object acceleration, m/s (bicycle model)
36 / 124	objCurvature	32	single	Object trajectory curvature, 1/m (bicycle model)
37 / 128	objLongVel	32	single	Object velocity, longitudinal part, m/s (Cartesian model)
38 / 132	objLatVel	32	single	Object velocity, lateral part, m/s (Cartesian model)
39 / 136	objLongAcc	32	single	Object acceleration, longitudinal part, m/s^2 (Cartesian model)
40 / 140	objLatAcc	32	single	Object acceleration, lateral part, m/s^2 (Cartesian model)
41 / 144	objWidth	32	single	Object width, m
42 / 148	objHeight	32	single	Object height, m
43 / 152	objConfidence	32	single	Object confidence
44 / 156	objLongCov	32	single	Covariance ellipse longitudinal size (relative to object)
45 / 160	objLatCov	32	single	Covariance ellipse lateral size (relative to object)
46 / 164	objCovHeading	32	single	Ellipse rotation in radians
47 / 168	objColor	8	int8 (enum)	0 = black, 1 = red, 2 = green, 3 = blue, 4 = cyan, 5 = magenta, 6 = yellow, 7-9 reserved, 10 - 17 light versions of 0-6, 18-max = undefined
48 / 169	objTransparency	8	int8	% (0= fully visible, 100 = invisible)
… Continues with a total of 'numObjects' objects. The next one begins at byte position 170.				

At the time of writing, the elements in Table 3 are fixed:

Table 3: Elements that currently are not updated

Signal Pos	Byte pos	Signal name	Value
3	16	numObjects	96
4	17	numTrails	0
5	18	coordSystem	0
7	23	egoVehType	3
8	24	egoWidth	2.5
9	28	egoLength	7.5
10	32	egoHeight	3.8
11	36	egoCSOffset	5.66
14	48	egoLongPos	0
15	52	egoLatPos	0
16	56	egoHeadingAngle	0
18	64	egoLatitude	57.706062
19	68	egoLongitude	11.939757
20	72	laneValid	1
21	73	laneLength	100
42	148	objHeight	1.4
46	164	covHeading	0
48	169	objTransparency	0