Covise

Notes

February 24, 2022

Contents

Part I.

Kernel

1. net

The net folder contains basic classes related to establishing connections between client and server using sockets.

1.1. Protocol

1.1.1. Overview

TCP Connection for regular messages, UDP Connection for UDP Messages, Optional with SSL Encryption (at least for regular messages, possibly for both). Also there has to exist a way to exchange data on the machine via shared memory.

There exist at least 2 distinct message types: Messages and UDP Messages. UDP Messages are regular messages with a stripped down header sent using the udp protocol (UDP Messages were introduced in May 2019, while regular messages exist since 1993).

The Default port used is 31000.

1.1.2. Message

1.1.2.1. Message Format

Defined in message

Each Message contains the following:

Name	Length	Comment
sender	3 byte	Sender of message, max 3 bytes
send_type	int	Sender Type, defaults to UNDEFINED, actual size depending on Architecture and Compiler. Should be 4 bytes on most modern systems, but could be 2. For a list of valid values, see Sender Types
type	int	Message Type, defaults to EMPTY, actual size depending on Architecture and Compiler. Should be 4 bytes on most modern systems, but could be 2. For a list of valid values, see Message Types
data		Bytes containing custom data

More specifically, the Header consists of 4 IEEE ints (16 bytes):

[0:3] sender

[4:7] senderType

[8:11] messageType

[12:15] dataLength - Length of data in bytes

1.1.2.2. Message Types

Defined in message_types

Name	ID	Comment
EMPTY	-1	
MSG_FAILED	0	
MSG_OK	1	
INIT	2	
FINISHED	3	
SEND	4	
ALLOC	5	
UI	6	
APP_CONTACT_DM	7	
DM_CONTACT_DM	8	
SHM_MALLOC	9	
SHM_MALLOC_LIST	10	
MALLOC_OK	11	
MALLOC_LIST_OK	12	
MALLOC_FAILED	13	
PREPARE_CONTACT	14	
PREPARE_CONTACT_DM	15	
PORT	16	
GET_SHM_KEY	17	
NEW_OBJECT	18	
GET_OBJECT	19	
REGISTER_TYPE	20	
NEW_SDS	21	
SEND_ID	22	
ASK_FOR_OBJECT	23	
OBJECT_FOUND	24	
OBJECT_NOT_FOUND	25	
HAS_OBJECT_CHANGED	26	

OBJECT_UPDATE	27	
OBJECT_TRANSFER	28	
OBJECT_FOLLOWS	29	
OBJECT_OK	30	
CLOSE_SOCKET	31	
DESTROY_OBJECT	32	
CTRL_DESTROY_OBJECT	33	
QUIT	34	
START	35	
COVISE_ERROR	36	
INOBJ	37	
OUTOBJ	38	
OBJECT_NO_LONGER_USED	39	
SET_ACCESS	40	
FINALL	41	
ADD_OBJECT	42	
DELETE_OBJECT	43	
NEW_OBJECT_VERSION	44	
RENDER	45	
WAIT_CONTACT	46	
PARINFO	47	
MAKE_DATA_CONNECTION	48	
COMPLETE_DATA_CONNECTION	49	
SHM_FREE	50	
GET_TRANSFER_PORT	51	
TRANSFER_PORT	52	
CONNECT_TRANSFERMANAGER	53	
STDINOUT_EMPTY	54	
WARNING	55	
INFO	56	
REPLACE_OBJECT	57	
PLOT	58	
GET_LIST_OF_INTERFACES	59	
USR1	60	
USR2	61	
USR3	62	
USR4	63	
NEW_OBJECT_OK	64	
NEW_OBJECT_FAILED	65	
NEW_OBJECT_SHM_MALLOC_LIST	66	
REQ_UI	67	

NEW_PART_ADDED	68	
SENDING_NEW_PART	69	
FINPART	70	
NEW_PART_AVAILABLE	71	
OBJECT_ON_HOSTS	72	
OBJECT_FOLLOWS_CONT	73	
CRB_EXEC	74	
COVISE_STOP_PIPELINE	75	
PREPARE_CONTACT_MODULE	76	
MODULE_CONTACT_MODULE	77	
SEND_APPL_PROCID	78	
INTERFACE_LIST	79	
MODULE_LIST	80	
HOSTID	81	
MODULE_STARTED	82	
GET_USER	83	
SOCKET_CLOSED	84	
NEW_COVISED	85	
USER_LIST	86	
STARTUP_INFO	87	
CO_MODULE	88	
WRITE_SCRIPT	89	
CRB	90	
GENERIC	91	
RENDER_MODULE	92	
FEEDBACK	93	
VRB_CONTACT	94	Anmeldung Client
VRB_CONNECT_TO_COVISE	95	
END_IMM_CB	96	
NEW_DESK	97	
VRB_SET_USERINFO	98	
VRB_GET_ID	99	
VRB_SET_GROUP	100	
VRB_QUIT	101	
VRB_SET_MASTER	102	
VRB_GUI	103	
VRB_CLOSE_VRB_CONNECTION	104	
VRB_REQUEST_FILE	105	
VRB_SEND_FILE	106	
VRB_CURRENT_FILE	107	
CRB_QUIT	108	

REMOVED_HOST	109	
START_COVER_SLAVE	110	
VRB_REGISTRY_ENTRY_CHANGED	111	
VRB_REGISTRY_ENTRY_DELETED	112	
VRB_REGISTRY_SUBSCRIBE_CLASS	113	
VRB_REGISTRY_SUBSCRIBE_VARIABLE	114	
VRB_REGISTRY_CREATE_ENTRY	115	
VRB_REGISTRY_SET_VALUE	116	
VRB_REGISTRY_DELETE_ENTRY	117	
VRB_REGISTRY_UNSUBSCRIBE_CLASS	118	
VRB_REGISTRY_UNSUBSCRIBE_VARIABLE	119	
SYNCHRONIZED_ACTION	120	
ACCESSGRID_DAEMON	121	
TABLET_UI	122	
QUERY_DATA_PATH	123	
SEND_DATA_PATH	124	
VRB_FB_RQ	125	
VRB_FB_SET	126	
VRB_FB_REMREQ	127	
UPDATE_LOADED_MAPNAME	128	
SSLDAEMON	129	
VISENSO_UI	130	
PARAMDESC	131	
VRB_REQUEST_NEW_SESSION	132	
VRBC_SET_SESSION	133	
VRBC_SEND_SESSIONS	134	
VRBC_CHANGE_SESSION	135	
VRBC_UNOBSERVE_SESSION	136	
VRB_SAVE_SESSION	137	
VRB_LOAD_SESSION	138	
VRB_MESSAGE	139	
VRB_PERMIT_LAUNCH	140	
BROADCAST_TO_PROGRAM	141	
NEW_UI	142	
PROXY	143	
SOUND	144	
LAST_DUMMY_MESSAGE	145	

1.1.2.3. Sender Types

Defined in message_types

Name	ID	Comment
UNDEFINED	0	
CONTROLLER	1	
CRB	2	
USERINTERFACE	3	
RENDERER	4	
APPLICATIONMODULE	5	
TRANSFERMANAGER	6	
SIMPLEPROCESS	7	
SIMPLECONTROLLER	8	
STDINOUT	9	
COVISED	10	
VRB	11	
SENDER_SOUND	12	
ANY		

1.1.3. UDP Message

1.1.3.1. UDP Message Format

Name	Length	Comment
type	int	Type of UDP Message. For a list of valid values, see UDP Message Types
sender	int	Sender of message, sender < 0 are invalid, sender 0 is the server and sender > 0 are clients
data		Bytes containing custom data

More Precisely, the header of the message consists of 2 IEEE Ints (8 Bytes), consisting of first the type, and then the sender. This is followed by the data. The entire message (consisting of both header and data) must not exceed the defined (system dependend) Write buffer size. See WRITE_BUFFER_SIZE in header file of covise_connect for the maximum total packet size. At the time of Writing, this is 393216 byte on CRAY systems, and 64000 byte on all other systems.

1.1.3.2. UDP Message Types

Name	ID	Comment
EMPTY	0	
AVATAR_HMD_POSITION	1	
AVATAR_CONTROLLER_POSITION	2	
AUDIO_STREAM	3	
MIDI_STREAM	4	

1.2. Classes

1.2.1. covise_connect

Handles the socket connection

1.2.2. message

Definition of standart message

1.2.3. message_types

Definition of message- and sender- types

1.2.4. udp_message_types

Definition of udp message- and sender- types

1.2.5. udpMessage

Definition of udp message

1.2.5.1. Trivia

- File is inconsitently named using CamelCase instead of underscores as seperators

1.2.6. udp_message_types

2. vrb

Part II.

Remarks

3. Protocols

- In order to ensure format compatibility, the code has to be compiled with a c++ compiler which implements 'int' as 32-bit value, correct ? Otherwise, the byte offsets in the protocol defined in net would not match the expectations. (e.g. in headder of udp messages)
  - int's are per definition IEEE int's with 4 bytes
- Why should the sender ID not exceed 3 bytes ? Looking at the protocol, there should be 4 bytes available in the protocol (both UDPMessage as well as regular Messages).