Manuscript-LIENa: A Novel distributed middleware for Tele-Surgery

Towards a design of middleware for distributed telesurgery system

# 1 Introduction

A modern operation is not merely the cooperation among surgeons, but rather among medical devices and humans or even just among devices based on the automatic holistic processing of all the available data in the future [Ref.01.surgical data for next-generation intervention]. In recent years, there is a certain number of robots used in the therapeutic area, aiming to assist or to accomplish surgical operation, such as Leonardo Da Vinci surgical robot, Corpath interventional vascular robot. However, the number of robot systems are still not enough for various types of surgeries. What's more, to equip with all kinds of medical robot systems costs too much for a hospital. Therefore, we are motivated to explore how to assemble current available medical resources such as medical equipment, software in the hospital to construct a flexible, extensive and sophisticated medical robot system.

As we all know, a robot system consists of four principal parts, which are hand, eye, brain, arm. The “hand” usually acts as the execution part of a robotic system; the “eye” will acquire real-time vision information, like different kinds of images; the “arm” is responsible for co-operating with the hand to achieve some positioning work, and the “brain” makes decisions and gives commands according to different types of input information. Noticing that the modern operating room is undergoing a period of significant changes, in which pervasive medical imaging devices, such as CBCT X-Ray Imaging system, Ultrasound imaging system, MRI system, can be acted as "eye" to serve a medical robot system. Meanwhile, the emerging ubiquitous surgical planning software like OSIRIX, MIMICS, with their customized navigation functionalities, can be acted as the brain of the medical robot system; besides, all the hospitals have Local Area Network environment which is a handy resource to build a sophisticated real-time distributed system for telesurgery. As (reference nature surgery data science) has mentioned, the lack of standardized protocols for complex data has become one of the challenges for next-generation interventions. Hence, to make the most of existing resources as we mentioned above, the primary objective of our study is to design a distributed communication component in order to create secure real-time communication session for each two predefined connected modules according to a given distributed topology network connection relationship in a medical robot system. And this component also has the ability to guarantee the efficiency and security of the real-time communication session and solve most system failure issues. Brief, we want to achieve that independent medical devices or software can be assembled together to serve as a medical robot system.

A considerable amount of literature has been published on the CPs system and the design of middleware of it. [system 02] has made a summary of this new class of engineered system, which offer close interaction between cyber and physical components. It is widely designed in different domains and application, which have common characteristics like cyber capability in every physical component, a high degree of automation…However, it poses some issues of CPs system, about security, resilience-reliability, QoS, real-time requirements. Facing these issues, to model and design real-world data sets, to specified components in the real world, to develop many aspects of CPs. Based on these conditions, a key to resolve this problem is to construct a middleware to meet blow demands: firstly, to standardize the message structure for facilitating decoding/encoding while real-time exchange; secondly, to manage sessions among different devices, such as establishing sessions, to close sessions, session diagnosis. Thirdly, using real-time scheduling algorithm to balance the bandwidth charge the in the multi-peer to peer real-time communication.

OpenIGTLink has focused on constructing a standardized, extensible network protocol for image-guided therapy environment, which gives a standard for communication among devices and software to share and transform information which has been summarized into three types: imaging, control, tracking. However, this protocol has not mentioned about dealing with real-time session management, such as the handshake, disengagement process between nodes, and most essentially: the scheduling strategy while network congestion and the failure handling procedure. Besides, its message protocol mainly depends on the functions of messages: system control, image, tracking. It cannot cover all the messages with the increasing of medical robots. The constant growth of robot systems will cause systems' message class to explode, which will make the size of the communication module extremely larger in the design of the application.

Therefore, we propose a customized peer to peer middleware named LiENa(Lightweight Unified Extensible Network Protocol for medical robotic system), a multi-language (python, C++) solution based on TCP/IP,…. In our middleware, we have proposed a new mechanism for decoding and encoding a variety of messages in the vastly distributed module in the medical robot system.

The overall structure of this paper takes the form of four chapters, including introductory chapter, following the second chapter of methods, where we present the fundamental design principals of LIENa, which compose of the network architecture, standardization of message set in this middleware, further, we will explain our session management mechanism during a multi-process operation and our algorithm for bandwidth balance. In the third part, we will present a series of experiments based on LIENa. Finally, a summary will be given.

# 2 Methods

We propose a peer-to-peer middleware named LIENa, which aims to provide a real-time message exchanging mechanism among various physical devices which can be act as a module of medical robot system.

我们将建立一个LiENa Physical device的社区，LiENa纳入该社区的仪器

In the middleware, we build an asynchronous peer-to-peer real-time communication architecture between every two networked modules in the medical robot system.

With a standard message set including session management such as connection fault detection, its recovery procedure related, the scheduling strategy while network congestion appears, the LIENa makes the real-time communication more safe, resilient, interoperable and reliable.

### Peer to Peer Middleware Design

We created an object-oriented component to provide a higher-level programming abstraction for the development of the distributed medical robot system to abstract over heterogeneity in the underlying infrastructure to promote interoperability and portability.

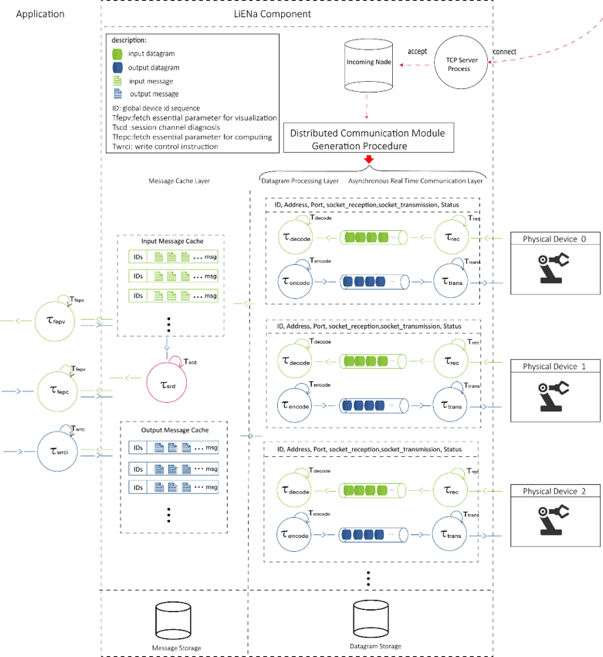
**Networking Paradigm**

Addressing the safety demand and particularity of the medical robot system, we chose the peer to peer communication architecture to realize the remote communication between the surgeon side and the patient side. As we all know, there are two general communication models: client-server model (CS) and peer to peer model (P2P). Traditional client-server architecture partitions task or workloads between the providers of a resource or service (server) from service requesters(clients). This protocol can accomplish real-time communication among multi-nodes, while once the server side occurs errors, the whole system is possible to break down.

Furthermore, we can only get the clients' status from the server side. The robotic system is a strict real-time system, the property of decentralization, the accessibility of any module in the peer to peer architecture is preferable. It is considered that quantitative distributed modules can extend to build a more comprehensive system. ­

**Middleware Architecture Overview**

To build the middleware LIENa, where different kinds of messages are exchanged in real time between physical devices module in a defined medical robot system, we designed an asynchronous real-time communication architecture and the message processing pipeline as is depicted in figure \*).



Figure\* overview of the middleware architecture

The middleware is divided into three part:

**Global Server:** Each middleware has a global server, once an incoming connection arrives, an input channel and a real-time reception task will be generated to receive all the messages using the socket return form the global server, at the meantime, a series of procedure will be launched to generate the distributed system module.

**Message Cache layer:** The layer stocks temporary input messages decoded from message processing layer and temporary output messages waiting to be encoded from application layer.

**Distributed Module layer:** In this layer, input message is processed from decoding data by decoding task(s) while output message is encoded into datagram by encoding task(s). To note that the number of decoding and encoding task is flexible, which can be adjusted according to the limit of bandwidth.

Based on our design, each distributed module in the pre-defined medical robot system has a global server. On the one side, once an incoming connection arrives, an input channel and a real-time reception task will be generated to receive all the messages using the socket return form the global server, at the meantime, a new client will be generated to connect to the TCP server of the incoming side , meanwhile, an output channel and a real-time transmission task will be generated to construct transmission chain. Through this procedure, each module of the distributed system creates two channels for the real-time asynchronous message communication with an incoming physical device. Based on this network architecture design, the new distributed module can be considered as a new distributed module, conveniently integrated into the distributed system architecture, without change the other real-time communication session.

Through the design of above software architecture, local medical robot component can open communication channels just by calling several functions in LIENa interface in application layer; Besides, it can exchange or share data with any other nodes in the system through standardized LIENa Message in the module of LIENa, hiding implementation details. What's more, developers can adjust the frequency of real-time tasks and augment or decline numbers of decoding or encoding tasks according to the limit of bandwidth, for balancing loads in communicating process among multi-nodes.

**Application layer:** The most upper layer, where different types of input message from cache layer are fetched for corresponding uses, such as session channel diagnosis, parameters for computing, image data for visualization, etc. Output messages such as session channel diagnosis feedback, commands from users are transmitted back to the cache layer.

**Object-Oriented Design of the Middleware**

To

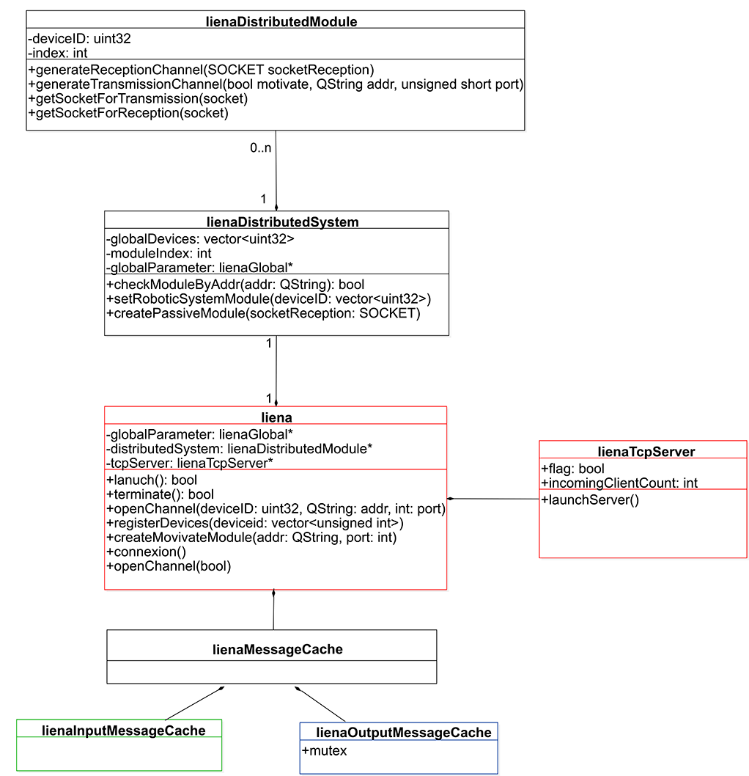


Fig.1. UML class diagram of the LIENa middleware

Figure \* gives an overview of LIENa software architecture, the­ package *LiENa* is composed of three principal modules, one of them is *LienaDistributedSystem* package, which manages LienaDistributedModule objects to communicate with the peripheral physical devices predefined to communicate with; another is *LienaMessageCache*, which creates the cache between system application and real-time transport layer. The other one, *LienaTcpServer,* defines a unique global TCP server, in which a local reception channel generation procedure for a *lienDistributedModule* will be triggered by an incoming client.

LienaDistributedModule design

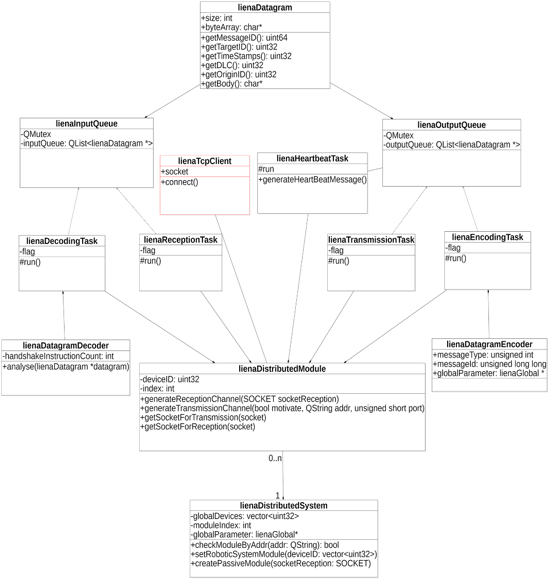
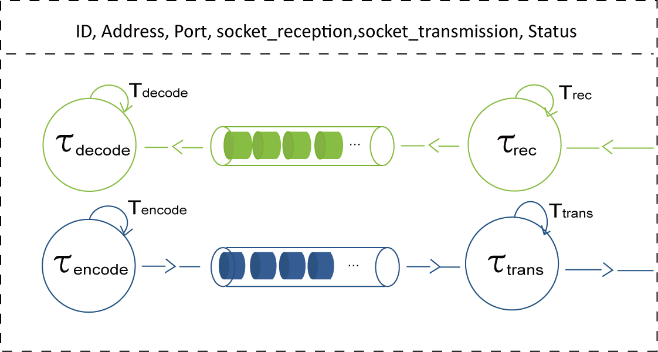
****

Fig.2. UML class diagram of LienaDistributedModule Module

Table 3 main classes in *lineaMessageCache* package

|  |  |  |
| --- | --- | --- |
| Function | Class | Description |
| real-time task | lienaDecodingTask | Realize decoding tasks |
| lienaEncodingTask | Realize encoding tasks |
| lienaReceptionTask | Recept real-time datagram |
| lienaTransmissionTask | transmit real-time datagram |
| lienaHeartbeatTask |  |
| datagram structure | lienaDatagram |  |
| lienaInputQueue |  |
| lienaOutputQueue |  |
| lienaDatagramDecoder | ­­­­ |
| lienaDatagramEncoder |  |
| others | lienaTcpClient |  |
| lienaDistributeModule |  |
| lienaDistributedSystem |  |

****

**x**

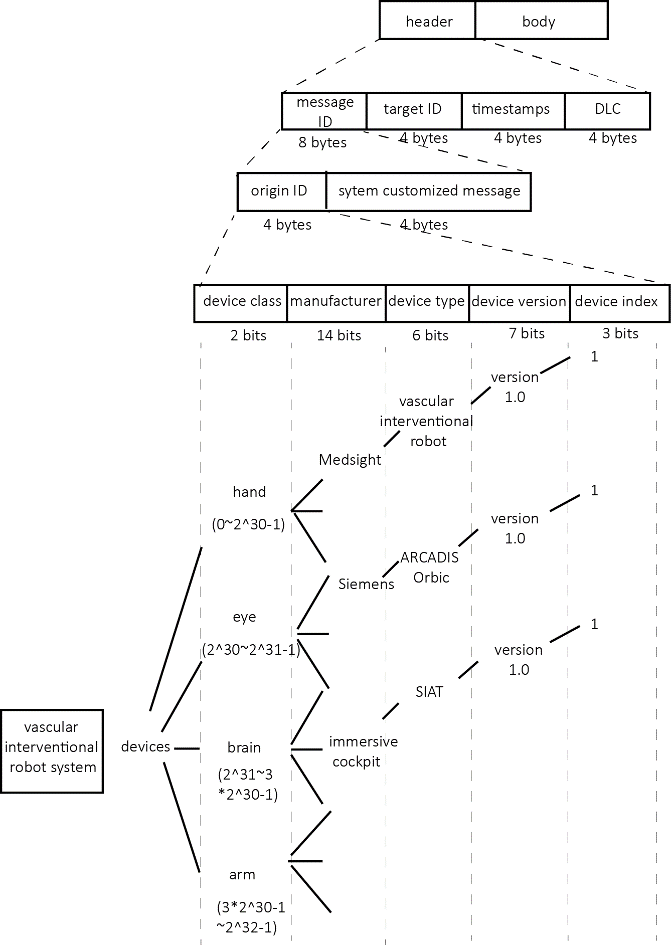
### LiENa Messaging Protocol

As we all know, the information of a physical device communicated in a medical robot system can be summarized into many types, such as control instructions, feedback information, global parameter configuration, real-time session management messages and so on. Therefore, how to encapsulate all that different information as a uniform format became a great challenge in this area. The primary purpose of our design is to provide a standardized mechanism for such a wide range of messages exchanged among physical devices which are underlying to become a distributed module of a medical robot system.

**Message Encapsulation**

We defined a fundamental data structure named *LienaMessage* which aim to encapsulate all type of information related to a physical de*vice.* Each *LienaMessage* can be split into a sequence of LienaDatagram, which is the basic unit of the middleware for communication among physical devices, and vice versa.

*LienaMessage*: defined in LiENa protocol, is composed of two parts. The first part is the meta-data part which includes the minimal but sufficient information to ensure that the message can be identified and applied in many functionalities in medical robot systems; And the second part is the information part which aim to encapsulating all the useful values of the message which defined by the system developer.



The meta-data part includes four necessary parameters: message type, target ID, time-stamps

*Message ID*: The message ID represents a unique message identifier of a given physical device, it is composed by the device id it belongs to and the customized message identifier for developers to implement. we proposed a rigorous classification method to assign a unique device identifier for each underlying physical device in LiENa protocol, each device identifier has a customized message set, which can be defined by the system developer of the physical device.

*Target ID***:** The target ID indicates the target device for the message to send and provides system double-check option and is widely performed in session manag­­ement functionalities.

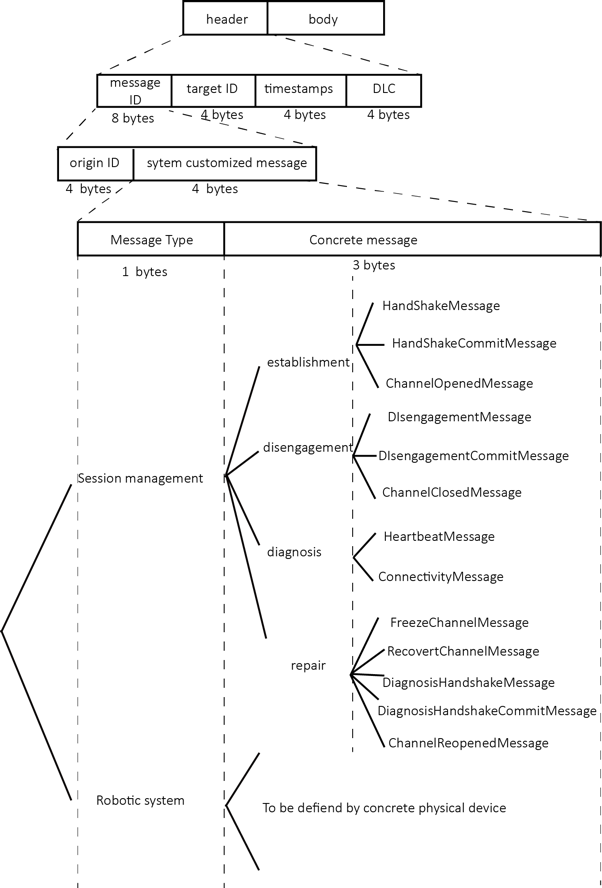
*Timestamps***:** records the time of generating the message for many usages such as: latency analysis, clock synchronization.

*DLC***:** The data length code calculates the length of the available message, giving parameters to adjust decoding frequency

*MSB*:

With the help of the meta-data part design, each message, communicating in the system, can be transformed to a distributed module's instant status after been unpacked, and a message sequence can be used to perform QoS functions.

The information part encapsulates entire information of a message based on the rule which has been predefined in the related XML file; the LIENa protocol defines the default length by the maximum length of the message of the physical devices registered.



Transform LienaCustomizedMessage 🡪 LiENaDatagram

**LiENaDatagram**:

Basic unit for communicate in LiENa Middleware

**DLC:** The data length code calculates the length of the available message, giving parameters to adjust decoding frequency.

For serializing heterogeneous data, we constructed a standard datagram structure, which is the basic unit of transmission. As the figure \* shows, we define the length of the header as 20 bytes which contains the data type, the target id, the origin id, the timestamps and DLC (data length code).

The header of a datagram is defined to cover all current medical device manufacturers, with abundant space to implement customized messages for developers. We set message id as the first parameter to let system perform an expeditious identification of the message to find related decode/encode method, where stocks the original id of device and system customized message for developers to implement. As it shows in Figure \*, the part of origin ID messages are divided by class, such as the system function: hand, eye, brain, arm, and a special class, which will be discussed in the following part. After classification, we set 14 bits space for manufacture id, which equals 2^14 (16384) manufacturers in each class, covering current manufacturers all over the world. And then, we categorize the message types into device type, indicating the function of this devices, device version, and device index, with assumption that there are maximum seven equal devices working together. The following parameters is target ID, which gives a destination for the datagram to send. Once the datagram is received, this information can be quickly decoded and identified by the target device. The parameter of timestamps adds time information for clock unification in the distributed system, and the parameter DLC gives the length of rest datagram to decode for adjusting decoding frequency.

­­­­

The body of the datagram depends on the type of messages; it comprises the information for exchange, chunking of big datagrams, such as intra/pre-operative imaging data. The length of the body is flexible; it can be adjusted to the message type and the real-time environment.

### Session Management

Given the characteristic of the communication aspect of a robotic medical system more precisely, a real-time distributed control system. We have focused our design on several system level aspect during the design of the middleware: first of all, depending on predefined system architecture, each device in a robotic system can find the right target devices to build a communication session with; secondly, we have to define the approach to establish/disengagement a real-time session between two physical devices which defined to be connected, and also the process to achieve the exchange of their own customized message set; thirdly, the mechanism to detect network failures and to provides a failure handling plan in the real-time communication process. Consequently, we integrated a series of session management dialog procedure, to realize all the previous aspects, including session establishment, session disengagement, session diagnosis and the recovery mechanism in our middleware.

**System Topology Registration**

As mentioned above, into LiENa Protocol, we assigned a unique device id for all medical robot module(physical device) . Based on this, we create a system topology file for each robotic system to represent its connectivities in all of its modules. This file will be loaded into the system primarily in a physical device to declare the system's architecture, through retrieving this file, each device can find its target devices, and they will exchange their own customized message set and save into its local database.

The message set of session management is a special set of the system customized message. Once the device is registered in our system, session management message will be automatically loaded in the device’s message database, preparing for manage session.

**Session establishment**

Motivation:

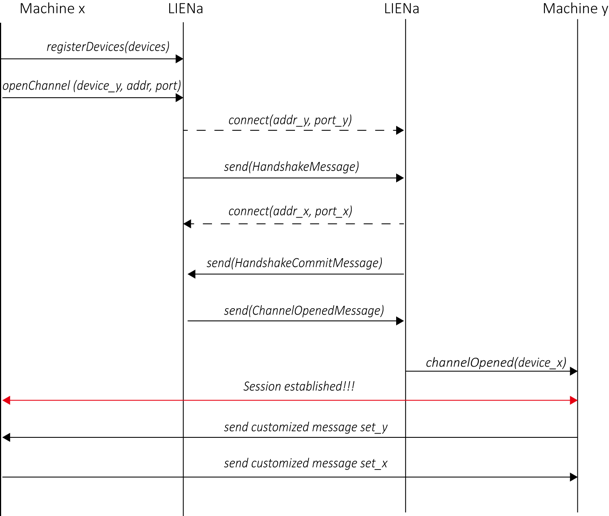


Figure \*. Establishing an open channel

The figure\* shows the process of establishing an open channel, which means creating a communication channel between two devices. Firstly, the machine x and machine y register in the LIENa middleware. The machine x commands *openChannel,* inputs the address and port of the target device—machine y. Then the LIENa will connect the LIENa of machine y and send a piece of *HandshakeMessage.* Once the machine y accepts this message in the LIENa model, it will link back to the machine x and submit a piece of *HandshakeCommitMessage* to the machine x. If the machine x accepts this message, it will transmit *ChannelOpened* to the machine y. Thus, the message transmitting channel is opened.

When the two devices connect at the first time, they will exchange their own message set with an XML file. Because the device id is encrypted in the high order of message ID as we mentioned above, each message ID is unique. After the messages exchange process, the message set of the other side will be stored in the local storage, which facilitates the obtention of encoding/decoding rule.

**Session Disengagement**

Motivation:

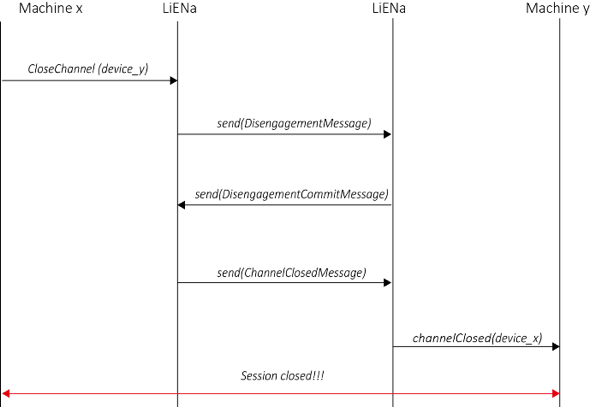


Figure \*. Open a session

The process of session disengagement is quietly similar to the session establishment. The machine x sends the *CloseChannel* commandto the LIENa, and the LIENa transmits the *DisengagementMessage* to the machine y. The LIENa of machine y receives this message, and after decoding this message, it will send *DisengagementCommitMessge* back. Once the machine x gets this message, it will send *ChannelClosedMessage* to the machine y. And the session will be closed after this process.

**Session Diagnosis**

In case one of the nodes in the real-time system loses connection, owing to network break or interruption of power supply, the heartbeat message is thus defined to detect the connection status in real-time. It is sent by each node to the target device in a particular time (such as 20ms) and decoded by the corresponding side. Once the system doesn’t receive *HeartBeatMessage* for a specific time, it will try to connect the opposite side again. If it failed to connect, all the related devices will stand-by, waiting for taking some measures, such as checking the physical status of the network, re-plugging network cable. After receiving user's commands, the system will send *DiagnosisHandshakeMessage,* the opposite side will send back *DiagnosisHandshakeCommitMessage,* and the channel will be reopened with broadcasting *ChannelReOpened.* At that time, all of the devices in the system will recover its connection and start to work.

Another consideration is to treat the load imbalance problem in the distributed system. When a session consumes too much bandwidth, the system will take some measures, for instance, several decoding tasks and encoding tasks will be generated to share the transmission task. (image data?)

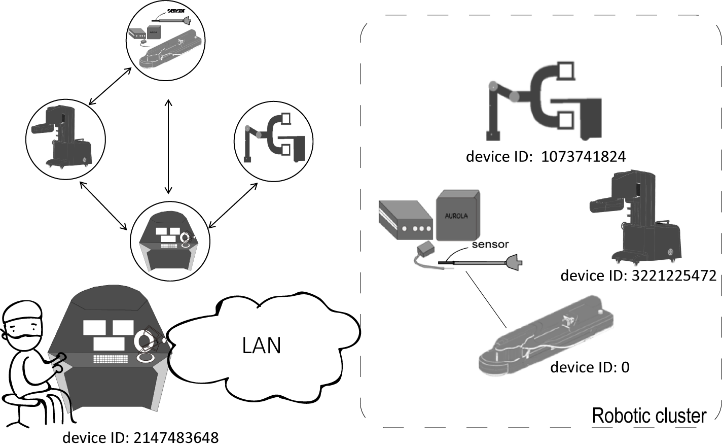
**Experiment**

**As**

**Case 1: Cerebrovascular Robot System, SIAT, Shenzhen, China:**

**Description**

We conducted experiments to evaluate the performance of LIENa middleware. One of them is the application in a remote vascular interventional robot system. With respect to interventional surgery, a master-slave system is popularly used, owning to its characteristic to separate doctors from X-ray environment. And in consequence the real-time communication plays an important role in the tele-surgery.

****

The experimental robot system composes of four basic devices: immersive cockpit (produced by SIAT, with a haptic manipulator Omega 3), interventional vascular robot with NDI magnetic tracking system (produced by Medsight, version 1), robotic arm (produced by Medsight, version 1) and CBCT system (ARCADIS Orbic, SIEMENS).

**Interventionist site:**

**Immersive cockpit:** provides interventionist an immersive environment to tele-operate interventional vascular surgery, with displaying patients’ information, analyzing pre-operation data and displaying the real-time X ray images data.

**Patient site:**

**Robotic arm grasper**: Robotic arm grasper is used to fix interventional robot. It can be adjusted by an application in the Linux or Windows operating system.

**Interventional robot**: It plays the major role in the interventional surgery as the execution part of interventional robot system. It is controlled by a Raspberry Pi (model 3B) and it can implement interventional surgical actions, such as guidewire advancing, guidewire rotating, catheter advancing and contrast agent injecting.

**CBCT system:** The CBCT system offers real-time image data during the surgery, aiming to assist surgeons to find the vascular lesion and make appropriate decision.

**2. message**

The interventional vascular robot system has two basic types of messages: the system control instruction and system status information. The system control instruction composes of the input messages of the robot system, receiving various types of commands to control the robot, including adjusting orientation and speed to different types of motors, controlling status of electric grippers, to realize the functions of vascular intervention. The system status information messages are the output messages, which are to send to the control console, transmitting the real-time system status, including the status of motors, sensors, the position information of the movement of the guidewire and the rest volume of contrast media.

**3. import message efficiency, failure handling case.**

discussion

Current commercial devices in the operating room have limited connectivity, especially for those from a different manufacturer, due to their different operating systems and supported platform.