KVM client development of OpenBMC to establish RFB protocol

原创 Lemon in the Rain Posted on 2024-11-09 13:36:04 Views:792 Collection 15 Likes 13

Copyright CC 4.0 BY-SA

Category Column: OpenBMC Article Tags: linux C++

OpenBMC This column includes this content 8 articles Subscribe to our column

KVM client development of OpenBMC to establish RFB protocol

1. Knowledge Supplement

- WebSocket protocol: a protocol for full-duplex communication over a single TCP connection. WebSocket makes data exchange between the client and the server much simpler, allowing the server to actively push data to the client. In the WebSocket API, the browser and the server only need to complete a handshake, and a persistent connection can be directly established between the two, and two-way data transmission can be carried out.
- RFB protocol: Remote Frame Buffer remote frame buffer protocol, a protocol used for remote access to the graphical user interface.
- Image transmission characteristics: The RFB protocol will draw the window in the video memory on the server side, and then transmit the image to the client. The client only needs to decode and display the obtained image.
- RFB Scope of application: RFB protocol is used for thin clients. Thin clients refer to reducing the burden on the client as much as possible, and most of the work is completed by the server.
- RFB protocol connection process: Like most protocols, it is also connected through the TCP/IP protocol suite.
  The first step is the handshake message, the purpose of which is to negotiate the protocol version and encryption method.
  The second step is the initialization message, which is mainly used for initialization messages between the client and the server.
  The last step is the normal protocol interaction, where the client can send messages as needed and then get a reply from the server.

2. How to get the image frame

1. Websocket connection
2. fb protocol information interaction to obtain image frame data
3. Fill canvas with image

3. WebSocket connection

The kvm front end creates a websocket and connects to the data transfer module; then the data transfer module creates a websocket and connects to the kvm back end to complete the establishment of the websocket connection channel.

```
  key: "_connect",
  value: function _connect() {
    Log.Debug(">> RFB.connect");
    Log.Info("connecting to " + this._url);

    try {
      // WebSocket.onopen transitions to the RFB init states
      this._sock.open(this._url, ['binary']);
    } catch (e) {
```

```
rfb = new RFB(target, tipTarget, 'wss://' + host + ':' + port + '/kvm/0', { 'isScaleViewport': true });
if (rfb) {
  console.log('rfb---------------')
}
```

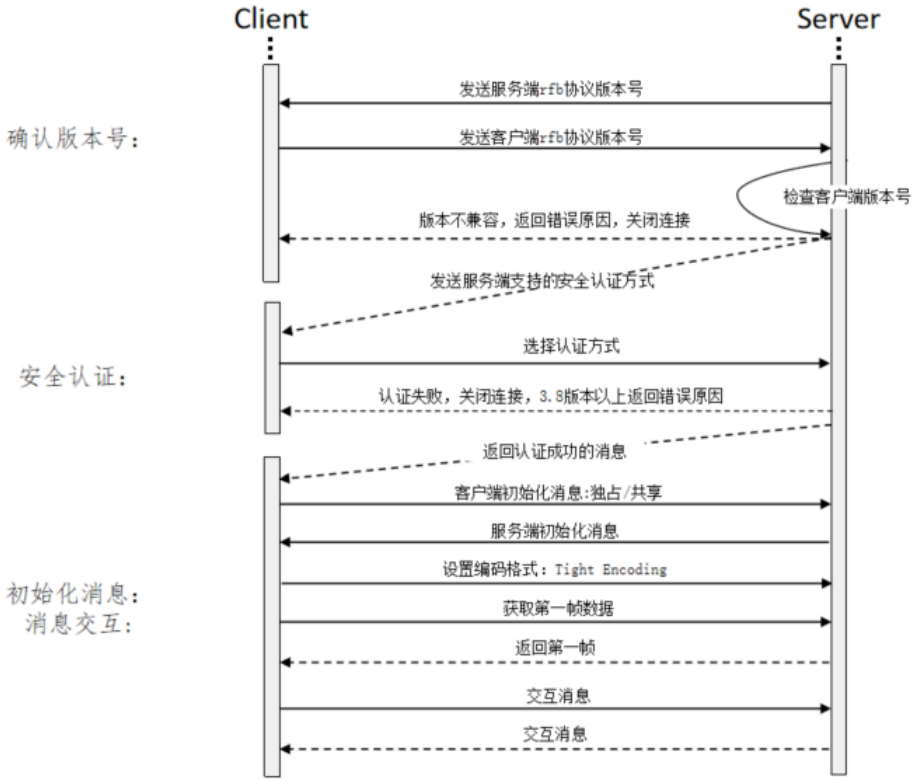
```
inline void requestRoutes(App& app)
{
    sessions.reserve(maxSessions);
    sessionsInfo.reserve(2);
    sessionsInfo.emplace(3,nullptr);
    sessionsInfo.emplace(4,nullptr);

    BMCWEB_ROUTE(app, "/kvm/0")
        .privileges({{"ConfigureComponents", "ConfigureManager"}})
        .websocket()
        .onopen([](crow::websocket::Connection& conn,
                   const std::shared_ptr<bmcweb::AsyncResp>&) {
```

Reference code:
webui\node_modules@novnc\novnc\core\ rfb .js
webui\node_modules@novnc\novnc\core\websock.js

4. RFB protocol interaction

The overall interaction process is as follows:



- Receive and send rfb version number:
  _handle_message()(line815) -> _init_msg()(line1746) -> _negotiate_protocol_vertion();

- The current version number is: RFB 003.008

No. of bytes	Value
12	"`RFB 003.003\n`" (hex 52 46 42 20 30 30 33 2e 30 30 33 0a)

or

No. of bytes	Value
12	"`RFB 003.007\n`" (hex 52 46 42 20 30 30 33 2e 30 30 37 0a)

or

No. of bytes	Value
12	"`RFB 003.008\n`" (hex 52 46 42 20 30 30 33 2e 30 30 38 0a)

- RFB protocol security authentication

AI generated projects 登录复制

```
安全认证流程及涉及接口：
确认主从权限：
   _negotiate_oem_magic() ;
接收安全认证方式类型列表：
   _negotiate_security();
选择安全认证方式： None
接收安全认证结果：
   _handle_security_result();

```

7.1.3 SecurityResult

The server sends a word to inform the client whether the security handshaking was successful

No. of bytes	Type	[Value]	Description
4	U32		status:
		0	OK
		1	failed
		2	failed, too many attempts [2]

version 3.8 onwards

If unsuccessful, the server sends a string describing

No. of bytes	Type	Description
4	U32	*reason-length*
reason-length	U8 array	*reason-string*

130	RSA-AES-256 Unencrypted Security Type
133	RSA-AES-256 Two-step Security Type

- rfb protocol information exchange

AI generated projects 登录复制

```
客户端初始化:
发送远程会话模式，独占或共享;
服务端初始化 _negotiate_server_init():
接收图像相关参数 _negotiate_server_init()
发送图像编码格式 clientEncodings()  Tight Encoding
消息交互 _normal_msg()(line1908) :
获取第一帧数据: _framebufferUpdate()(line2182) ;

```

7.3.1 ClientInit

No. of bytes	Type	Description
1	`U8`	*shared-flag*

7.3.2 ServerInit

After receiving the *ClientInit* message, the server sends a *Se*
framebuffer, its pixel format and the name associated with

No. of bytes	Type	Description
2	`U16`	*framebuffer-width*
2	`U16`	*framebuffer-height*
16	`PIXEL_FORMAT`	*server-pixel-format*
4	`U32`	*name-length*
name-length	`U8` array	*name-string*

5. Canvas image filling

- Create a canvas node:
  document.createElement()
- Node binding:
  If the Websocket connection is successfully established, the canvas node will be bound to noVNC_container; if the connection is cancelled, the corresponding canvas node will be removed;
- Image filling:
  _framebufferUpdate ()(line2182)
  display.flip()(line256) drawImage();

```
var target = angular.element(document.querySelector('#noVNC_container'))[0];
var tipTarget = angular.element(document.querySelector('#kvmTips'))[0];

try {
  rfb = new RFB(target, tipTarget, 'wss://' + host + ':' + port + '/kvm/0', { '
  if (rfb) {
    console.log('rfb----------------')
```

kvm-controller-window.js

```
  key: "_connect",
  value: function _connect() {
    Log.Debug(">> RFB.connect");
    Log.Info("connecting to " + this._url);

    try {
      // WebSocket.onopen transitions to the RFB init states
      this._sock.open(this._url, ['binary']);
    } catch (e) {
      if (e.name === 'SyntaxError') {
        this._fail("Invalid host or port (" + e + ")");
      } else {
        this._fail("Error when opening socket (" + e + ")");
      }
    } // Make our elements part of the page

    this._target.appendChild(this._screen);

    this._cursor.attach(this._canvas);

    this._refreshCursor(); // Monitor size changes of the screen
    // FIXME: Use ResizeObserver, or hidden overflow
```

rfb.js

about Us Careers Business Cooperation Seeking coverage 400-660-0108 kefu@csdn.net Online Customer Service Working hours 8:30-22:00

Public Security Registration Number 11010502030143 Beijing ICP No. 19004658 Beijing Internet Publishing House [2020] No. 1039-165

Commercial website registration information Beijing Internet Illegal and Harmful Information Reporting Center Parental Control

Online 110 Alarm Service China Internet Reporting Center Chrome Store Download Account Management Specifications

Copyright and Disclaimer Copyright Complaints Publication License Business license

©1999-2025 Beijing Innovation Lezhi Network Technology Co., Ltd.