Wolfpack Matrix Command Protocol Fix

Issue Summary

The Wolfpack matrix switching tests were experiencing command timeouts despite successful TCP connections to port 23. The connection test passed, but all switching commands timed out after 10 seconds without receiving any response from the device.

Root Causes Identified

1. Missing Line Endings (Primary Issue)

Problem: Commands were sent without proper line termination (`\r\n`)
- The code was sending raw commands like `1>7.` without carriage return and line feed
- Telnet/TCP protocol (port 23) requires commands to end with `\r\n` (CR+LF)
- Without proper line endings, the Wolfpack device doesn't recognize the command as complete and never responds

Evidence from Research:
- Wolfpack devices use Telnet protocol on port 23
- Standard Telnet protocol requires `\r\n` line endings
- API documentation confirms commands must be properly terminated

2. Inconsistent Command Format (Secondary Issue)

Problem: Test code used different command format than main routing code
- Test code: `1>7.` (input>output)
- Main routing code: `1X2.` (inputXoutput)
- Research shows Wolfpack devices support multiple formats depending on model:
- Format 1: `[input]X[output].` (most common)
- Format 2: `[input]V[output].` (some models)
- Format 3: `[input]>[output].` (less common)

Decision: Standardized on `X` format as it's used in the main routing API and is most widely documented

3. Inadequate Response Handling

Problem: Limited handling of various response patterns
- Only checked for exact "OK" or "ERR" strings
- Didn't handle cases where device might:
- Echo the command back
- Send response without explicit OK/ERR
- Close connection after sending response
- Send multi-line responses

Changes Made

File: `src/app/api/tests/wolfpack/switching/route.ts`

1. Added Line Endings to Commands

```
// Before:
socket.write(wolfPackCommand)

// After:
const commandWithLineEnding = wolfPackCommand + '\r\n'
socket.write(commandWithLineEnding)
```

2. Changed Command Format

```
// Before:
const command = `${input.channelNumber}>${output.channelNumber}`

// After:
const command = `${input.channelNumber}X${output.channelNumber}`
```

3. Enhanced Debug Logging

- Added hex dump of sent commands
- Log all received data with hex representation
- Track response accumulation
- Log timeout details with partial responses

4. Improved Response Handling

- Added `responseReceived` flag to prevent duplicate resolutions
- Handle socket close events with partial data
- Accept responses without explicit OK/ERR if substantial data received
- Better error messages showing what was actually received

File: `src/app/api/matrix/route/route.ts`

1. Added Line Endings to TCP Commands

```
const commandWithLineEnding = command + '\r\n'
client.write(commandWithLineEnding)
```

2. Increased Timeout

- Changed from 5 seconds to 10 seconds for consistency
- Allows more time for device response

3. Enhanced Response Handling

- Track response accumulation
- Handle connection close with partial data
- Better logging of timeout scenarios

4. Updated UDP Commands

- Added `\r\n` to UDP commands as well
- Changed exact match to `includes()` for more flexible response checking

Technical Details

Telnet Protocol Requirements

- Port 23 is standard Telnet port
- Commands must end with `\r\n` (ASCII 13 + 10)
- Device may echo commands back
- Responses typically end with `\r\n`

Wolfpack Command Format

Standard format: `[input]X[output].\r\n`
- `[input]` : Input channel number (1-32)
- `X` : Delimiter (some models use `V` or `>`)
- `[output]` : Output channel number (1-32)
- `.` : Command terminator
- `\r\n` : Line ending (required for Telnet)

Example: `1X7.\r\n` routes input 1 to output 7

Response Patterns

Wolfpack devices may respond with:
1. `OK\r\n` - Command successful
2. `ERR\r\n` - Command failed
3. `Error: [message]\r\n` - Specific error
4. Echo of command followed by response
5. Silent success (connection closes)

Testing Recommendations

1. **Run the switching test** from the System Admin page
2. **Check server logs** for debug output:
   - Look for `[DEBUG]` prefixed messages
   - Verify commands are sent with `\r\n`
   - Check hex dumps of sent/received data
   - Confirm responses are being received
3. **If still failing**, check logs for:
   - What exact bytes are being sent (hex dump)
   - What responses (if any) are received
   - Whether device is echoing commands
   - Connection close timing
4. **Alternative command formats** to try if needed:
   - `1V7.\r\n` (V delimiter)
   - `1>7.\r\n` (> delimiter)
   - Without period: `1X7\r\n`

Expected Behavior After Fix

1. **Connection**: Successfully connects to 192.168.5.100:23 ✓ (already working)
2. **Command Send**: Sends `1X7.\r\n` with proper line endings

3. **Response**: Receives `OK\r\n` or similar response
4. **Test Result**: Shows success with response details
5. **Logs**: Debug output shows hex dumps and response tracking

Rollback Plan

If issues persist, the previous behavior can be restored by:
1. Removing `\r\n` from commands
2. Reverting command format to `>`
3. Simplifying response handling

However, based on Telnet protocol standards and Wolfpack documentation, the current fix should resolve the timeout issues.

References

- Wolfpack API Commands Documentation
- Telnet Protocol RFC 854
- RS232/TCP Control Documentation for Wolfpack Matrix Systems
- HDBaseT Matrix Control Standards