CBIT Architecture Document

Version 1.0

Ronak Malik
Cypress Woods High School

August 2020

Table of Contents

Introduction

This document outlines the design implementation for the Circuit Board Inspection Table[CBIT] system. This system allows for the remote control of a 4-axis inspection mill and is able to stream high quality circuit board video in real-time for viewing by a client. Additionally, both the mill and the viewing computer are geographically separated and behind firewalls.

The system was tested using a Sherline 4-Axis Mill affixed with a PTZ Optics IP Camera located at a remote vendor and the client located at the Johnson Space Center.

1.1 Scope

This document applies to the design and implementation of the Mill Controller Program, Odroid Controller Program, Client Controller Program, and Bounce Server Program.

1.2 Definitions, Acronyms, and Abbreviations

CBIT – Circuit Board Inspection Table; A mill with up to 4 axes driven by LinuxCNC with an attached camera.

Mill Computer – The computer connected to the mill that is running the MillControllerProgram software.

Client Computer – The computer receiving video and sending control commands to the Mill Computer through the Client Controller Program.

Odroid – A single board computer that was added for this project and is connected to the Mill Computer via a LAN connection. The inspection camera is connected to this computer.

Bounce Server – A server that located outside of both firewalls that accepts incoming connections.

LinuxCNC – An open source Linux driver that controls the mill hardware.

RTAI – An open source real time application interface for Linux that allows program execution within strict time constraints.

Bounce Server Program – The software developed for this project run on the Bounce Server that relays data between two incoming connections.

Mill Controller Program – The software developed for this project run on the Mill Computer that handles control commands and interfaces with LinuxCNC.

Odroid Controller Program – The software developed for this project run on the Odroid that handles streaming video to the remote client.

Client Controller Program – The software developed for this project run on the Client Computer that receives and displays video, and sends user commands to the Mill Computer.

Mill Firewall – The firewall located at the remote manufacturing facility; the mill computer is behind this firewall.

Client Firewall – The firewall located at the Johnson Space Center; the client computer is behind this firewall.

NAT – Network Address Translator; A feature of the firewall that routes external connections to internal computers.

Clients – Generically used if a feature applies symmetrically to both the Mill Computer and the Client Computer.

Architectural Goals and Constraints

The main constraint with this system is the need to establish a connection between two peers without interfering with the firewall of either client, whether the interference be solicited such as active port forwarding or not. Traversal through the NAT was achieved by using a relay server (or bounce server) located outside both firewalls which accepts incoming connections, and routes data from each side to the corresponding client.

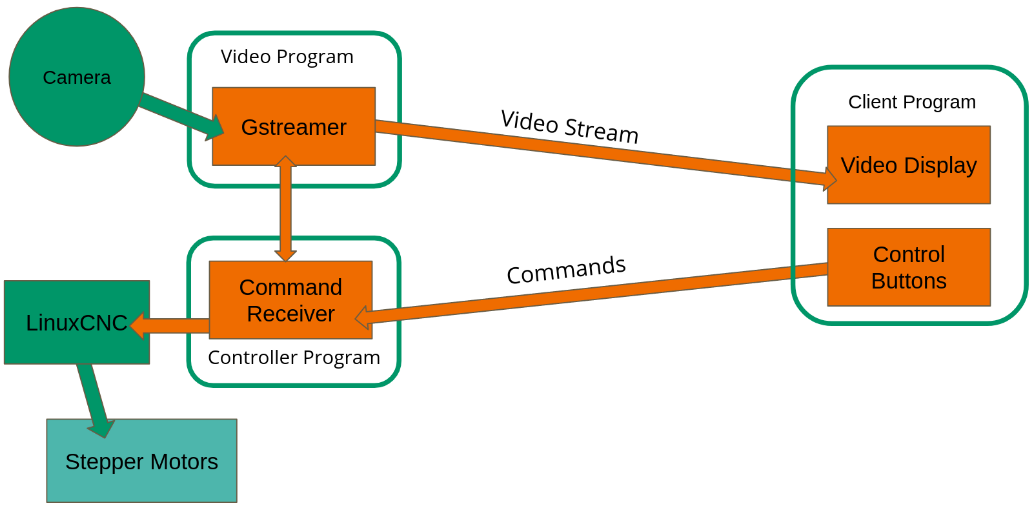
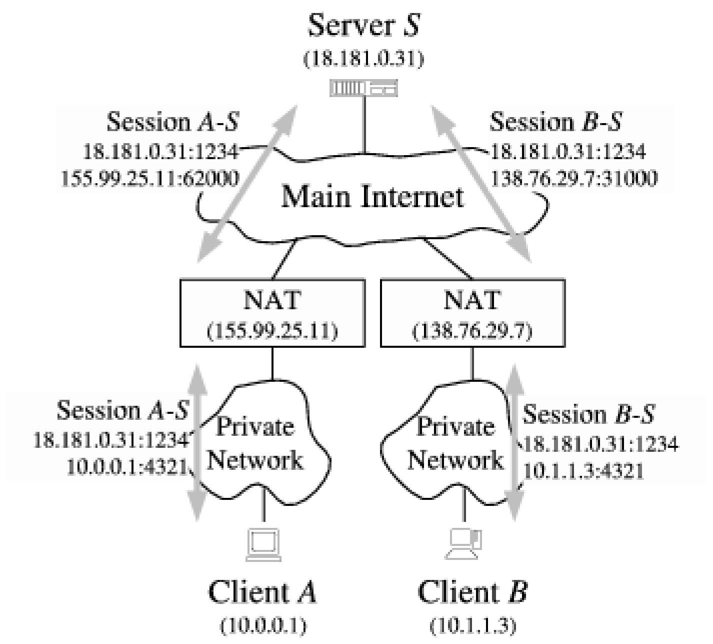


Image 1: Shows the effective transfer of data from the perspective of the two clients.



Source: Bryan Ford, M.I.T, p2pnat

Image 2: Interaction of clients with bounce server

By using the external bounce server, we can link two connections in software so that the once the connection to the bounce server is established, the clients can treat it as a normal connection.

Additionally, the skeleton of this system can be used to transfer any arbitrary data securely from any two computers behind secure firewalls.

Bounce Server Protocol

1.1 Identification

Immediately after connecting, the bounce server will send a version string to the client, which will look something like this: 4-BounceServer_1.0 This version string follow the format: n-BounceServer_{VERSION_NUMBER} Where n is the maximum number of digits, in hex, the channel number can be.

The client should wait to receive this string before sending any data.

1.2 Authentication

The next bytes that the bounce server expects to receive from the client is the authentication string. As of BounceServer 1.0, there is no longer a limit on the length of the authentication string.

This string is preset by the user of the bounce server to ensure that random connections are not permitted. The client has, by default, 5 seconds to send the authentication string before it is disconnected by the bounce server. This timeout is configurable by the user.

1.3 Channel Setting

The next n bytes are expected to be the client's channel number as a hex string. It is not required that the client pad the channel number with 0s, but it is recommended to ensure that the requested channel is correct.

1.4 Data Transfer

Once the authentication string and channel number have been sent, the client is free to send data to the bounce server, where it will blindly forward the data to all other connections on the same channel. Depedning on the configuration of the bounce server, it may also echo data back to the sending channel.

An example handshake between the bounceserver and a client would be as follows:

BounceServer -> Client1: "4-BounceServer_1.0"
Client1 -> BounceServer: "hello"
Client1 -> BounceServer: "00ac"
BounceServer -> Client2: "4-BounceServer_1.0"
Client2 -> BounceServer: "hello"
Client2 -> BounceServer: "00ac"
Client1 -> BounceServer: DATA

BounceServer -> Client2: DATA

First, Client1 sends the authentication message "hello", then requests to join channel 172. Then Client2 sends the authentication message and requests to join channel 172. Now all data sent from Client1 is forwardrd to Client2, and vice versa

Logical View

1.1 Overview

The entire system can be architecturally split into four packages: The Mill Controller Program, The Client Controller Program, The Odroid Controller Program, and the Bounce Server. The Bounce Server allows the two clients to communicate without interference from their respective firewalls and NATs, and the Mill Controller Program and Client Controller Program accept commands and send commands respectively to control the mill to see video.

1.2 Architecturally Significant Design Packages

GStreamer – An open source video library that powers the video streaming. LGPL license.

Netty – An open source networking library that allows for async networking. Apache License.

ml.dent.net – The networking abstraction made for this project that allows arbitrary data writes to the network, handles establishing a proxy, connecting to the bounce server, authenticating, and ensuring two-hop connectivity to the MillProgram. The rest of system in each client is oblivious to what connection is currently active and how it was made, and is only notified if the connection is dropped or if a data write fails. This framework is powered by Netty for asynchronous, event-driven networking.

ml.dent.video – The video package made for this project that reads from the video source, encodes and packages it if necessary into a network compatible form, and sends it over the network. The video reading and encoding/decoding is powered by GStreamer, while the networking is built on top of the ml.dent.net framework.

Deployment View

The CBIT Host software needs to run on a computer connected to a mill controllable by LinuxCNC, and the Client Controller can run on any computer that can run java. Both clients need to be network capable.

The Bounce Server must be deployed to a server that is not behind the same firewall of either client for the system to function properly. The server must also have a single port opened to allow incoming connections into the server. This port can be anything not currently in use by the server. Both client softwares support the use of an HTTPS proxy, so setting up a web proxy to allow for anonymous SSL encrypted data transfer is possible.

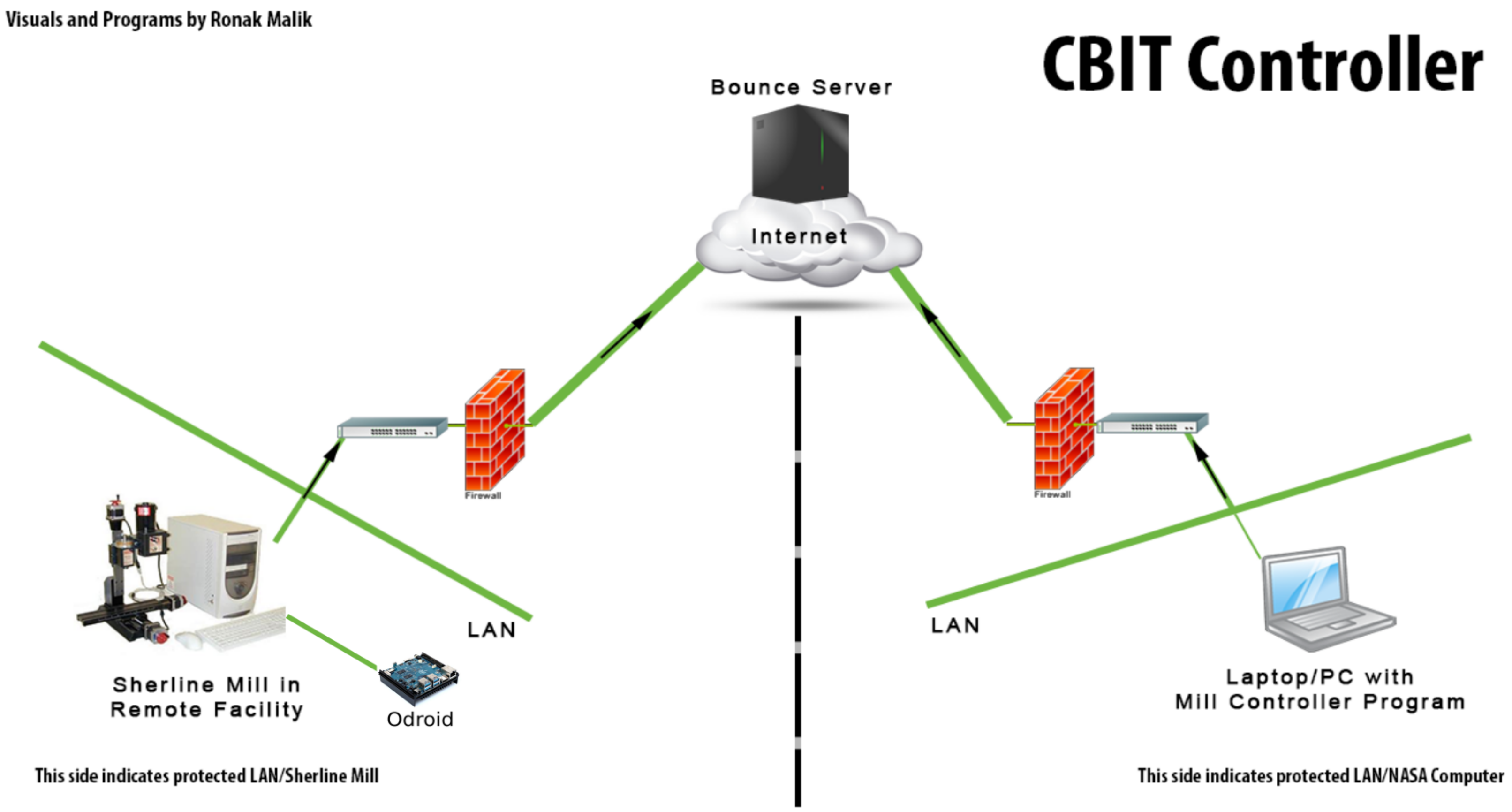

Visuals and Programs by Ronak Malik
CBIT Controller
Bounce Server
Internet
Firewall
Firewall
LAN
LAN
Sherline Mill in Remote Facility
Odroid
Laptop/PC with Mill Controller Program
This side indicates protected LAN/Sherline Mill
This side indicates protected LAN/NASA Computer

Image 3: Shows deployment of bounce server, and network representation of Mill Computer, Odroid, and Client Computer

Code Links

Code is currently available on GitHub

https://github.com/BeyondPerception/BounceServer

https://github.com/BeyondPerception/MillClientController

https://github.com/BeyondPerception/SherlineClientController