Lan Conference Application - Complete Documentation

Table of Contents

---

1. Introduction

LAN Conference Application is a comprehensive, local area network (LAN) based video conferencing solution built with Python and PyQt5. It provides enterprise-grade features including real-time video/audio communication, screen sharing, collaborative whiteboard, file transfer, and gesture recognition - all running on your private network without relying on external servers.

Key Highlights

- **Privacy-First**: All communication stays within your local network
- **Password Protected**: Secure access with auto-generated server passwords
- **Feature-Rich**: Video, audio, screen sharing, whiteboard, file transfer, and more
- **Multi-User Support**: Up to 50+ simultaneous participants
- **Cross-Platform**: Works on Windows, macOS, and Linux

Use Cases

- Corporate meetings and presentations
- Remote team collaboration
- Educational lectures and workshops
- Private video calls without internet dependency
- Collaborative brainstorming sessions

---

2. System Architecture

Client-Server Model

The application follows a centralized server architecture where: - **Server**: Acts as a relay hub for all communications - **Clients**: Connect to the server and communicate through it

Communication Protocols

TCP Connections (Reliable)

- **Port 9000**: Control messages (chat, user management, commands)
- **Port 9001**: Screen sharing data
- **Port 9002**: File transfers

UDP Connections (Real-time)

- **Port 10000**: Video streaming (prioritizes speed over reliability)
- **Port 11000**: Audio streaming with mixing capabilities

Data Flow Architecture

```
Client A                    Server                    Client B
   |                           |                          |
   |---Video Frame (UDP)----->|                          |
   |                           |----Mixed Video (UDP)------>|
   |                           |                          |
   |<--Audio Stream (UDP)-----|<---Audio Stream (UDP)------|
   |                           |                          |
   |---Chat Message (TCP)---->|----Chat Message (TCP)----->|
   |                           |                          |
```

---

3. Installation & Setup

System Requirements

Minimum Requirements: - Python 3.7 or higher - 2 GB RAM - 100 MB disk space - Network adapter with LAN connectivity - Webcam (for video features) - Microphone and speakers (for audio features)

Recommended Requirements: - Python 3.8+ - 4 GB RAM or more - 500 MB disk space - HD webcam (720p or higher) - Dedicated audio interface

Installation Steps

1. Install Python Dependencies

```
# Install all required packages
pip install -r requirements.txt
```

2. Platform-Specific Setup Windows:

```
# PyAudio usually installs directly
pip install pyaudio
```

macOS:

```
# Install PortAudio first
brew install portaudio
pip install pyaudio
```

Linux (Ubuntu/Debian):

```
# Install system dependencies
sudo apt-get update
sudo apt-get install portaudio19-dev python3-pyaudio
pip install -r requirements.txt
```

3. Verify Installation

```
# Check all imports
python -c "import cv2, PyQt5, pyaudio, mss, mediapipe; print('All dependencies installed!')'
```

Starting the Application

Start the Server (Host Machine)

```
python server3.py
```

Expected Output:

```
==================================================
 SERVER PASSWORD: A3X9
==================================================
[TCP] Control server listening on 0.0.0.0:9000
[VIDEO] Forwarder listening on UDP 10000
[AUDIO] Receiver listening on UDP 11000
[SCREEN] Relay listening on TCP 9001
[FILE] Server listening on TCP 9002
```

Important: Note down the 4-character password displayed!

Start the Client (All Participants)

```
python client4.py
```

---

4. Core Features

4.1 Password Authentication

Purpose: Prevents unauthorized access to your conference.

How It Works: 1. Server generates a random 4-character alphanumeric password on startup 2. Password is displayed in server console (e.g., `A3X9`, `K7P2`) 3. All clients must enter this password to join 4. Invalid attempts are logged and rejected

Security Features: - Password changes every time server restarts - Failed login attempts are logged - Immediate disconnection on authentication failure - No password storage or transmission in plain sight beyond initial display

4.2 Video Conferencing

Features: - Real-time video streaming at 20 FPS - Automatic video layout adjustment based on participant count - JPEG compression for efficient bandwidth usage - Picture-in-picture mode for own video feed

Technical Details: - Resolution: 320x240 (configurable) - Codec: JPEG with 80% quality - Bandwidth: ~200-400 KB/s per stream - UDP-based transmission for low latency

Usage: 1. Click " Start Video" button 2. Grant camera permissions if prompted 3. Video appears in main conference area 4. Click " Stop Video" to disable

Video Layout Modes: - **1 User**: Full-screen video (1050x700) - **2 Users**: Large main + small PiP (850x600 + 280x210) - **3 Users**: Main video + 2 thumbnails - **4+ Users**: Grid layout (340x280 tiles)

4.3 Audio Communication

Features: - High-quality audio streaming (16kHz sampling rate) - Real-time audio mixing on server - Packet Loss Concealment (PLC) for smooth playback - Echo cancellation capabilities

Technical Details: - Sample Rate: 16,000 Hz - Channels: Mono (1 channel) - Format: 16-bit PCM - Chunk Size: 256 samples (~16ms latency) - Buffer: 10 packets maximum

Audio Mixer Algorithm: The server uses intelligent audio mixing:

```
# Pseudo-code
for each_target_client:
    exclude_own_audio()
    mix_all_other_streams()
    apply_normalization()
    send_mixed_audio()
```

Usage: 1. Click " Start Audio" button 2. Grant microphone permissions if prompted 3. Speak normally - audio is automatically transmitted 4. Click " Stop Audio" to mute

4.4 Screen Sharing

Features: - Share entire screen with all participants - Presenter-viewer model (one presenter at a time) - Optimized compression for smooth performance - Expandable/collapsible viewer panel

Technical Details: - Resolution: 800x450 (scaled from full screen) - Frame Rate: 10 FPS - Compression: JPEG at 50% quality - Protocol: TCP for reliability - Bandwidth: ~300-600 KB/s

Presenter Mode: 1. Click " Share Screen" button 2. Server validates no active presenter exists 3. Screen capture begins automatically 4. All viewers receive notification 5. Click " Stop Sharing" to end

Viewer Mode: 1. Click " View Screen" when presenter is active 2. Screen panel expands automatically 3. Real-time screen updates appear 4. Click " Stop Viewing" to close

Conflict Resolution: - Only one presenter allowed at a time - New presenter automatically replaces previous one - Viewers notified of presenter changes

4.5 Chat System

Features: - Group chat (broadcast to all) - Private/direct messaging - System notifications - Message history

Message Types:

1. **Group Chat** (default)
   - Visible to all participants
   - Sender name displayed
   - Instant delivery
2. **Private Chat**
   - One-to-one messaging
   - Click "Switch to Direct" → Select user
   - Both sender and recipient see messages
   - Marked with icon
3. **System Messages**
   - User join/leave notifications
   - File transfer notifications
   - Gesture broadcasts
   - Error messages

Usage: 1. Type message in input field 2. Press Enter or click "Send" 3. For private chat: - Click "Switch to Direct" - Select recipient from list - Send messages normally - Click "Switch to Direct" → "Group Chat" to return

4.6 Collaborative Whiteboard

Features: - Real-time collaborative drawing - Multiple drawing tools - Multi-user cursor tracking - Synchronized state across all clients

Drawing Tools:

1. **Pen** (default)
   - Freehand drawing
   - Customizable width and color
   - Smooth line rendering
2. **Circle**
   - Click and drag to draw circles
   - Center point = first click
   - Radius = drag distance
3. **Rectangle**
   - Click and drag for rectangles
   - First corner = first click
   - Opposite corner = release point
4. **Line**
   - Straight line tool
   - Start point = first click
   - End point = release point

Whiteboard Actions:

- **Undo**: Remove last drawn element
- **Clear**: Erase entire canvas
- **Save State**: Automatically synchronized to server

Multi-User Cursors: - Each user has a colored cursor indicator - Real-time position updates - Username label follows cursor - Helps coordinate collaborative drawing

Technical Implementation:

```
// Each drawing action is:
{
  "id": "uuid-v4",
  "type": "stroke|circle|rect|line",
  "points": [...],
  "color": "#000000",
  "width": 3,
  "timestamp": 1234567890
}
```

Usage: 1. Click “ Whiteboard” button 2. Select tool from toolbar 3. Draw on canvas 4. All participants see changes instantly 5. Click “ Hide Board” to return to video view

4.7 File Sharing

Features: - Upload files to shared server storage - Download files from server - Automatic notifications on file upload - Support for any file type

Technical Details: - Storage Location: `server_files/` directory on server - Max File Size: No hard limit (depends on available disk space) - Transfer Protocol: TCP for reliability - Chunk Size: 64 KB for efficient streaming

File Transfer Process:

Upload Flow:

```
Client → [Select File] → Upload to Server → Server Stores
                                ↓
                    Broadcast notification to all clients
                                ↓
                    File appears in Files tab for all users
```

Download Flow:

```
Client → [Click Download] → Request from Server → Receive File
                                ↓
                    Save to: ~/Downloads/ConferenceFiles/
```

Usage:

To Share a File: 1. Go to " Files" tab 2. Click " Upload File" 3. Select file from dialog 4. Wait for upload confirmation 5. All users receive notification

To Download a File: 1. Go to " Files" tab 2. Find file in list 3. Click " Download" button 4. File saves to `~/Downloads/ConferenceFiles/` 5. Success notification appears

File Card Display:

```
 presentation.pdf
From: John | Size: 2.45 MB
[ Download]
```

4.8 Gesture Recognition

Features: - AI-powered hand gesture detection - Real-time gesture broadcasting - Animated emoji feedback - Five supported gestures

Supported Gestures:

1. **Thumbs Up**
   - Detection: Thumb extended up, other fingers folded
   - Use Case: Agreement, approval
2. **Peace Sign**
   - Detection: Index and middle fingers extended in V-shape
   - Use Case: Victory, peace

3. **Wave**
   - Detection: All four fingers extended
   - Use Case: Greeting, goodbye
4. **Heart**
   - Detection: Two hands forming heart shape with thumbs and index fingers
   - Use Case: Love, appreciation
5. **Clap**
   - Detection: Two hands close together, fingers extended
   - Use Case: Applause, celebration

Technical Details: - Engine: Google MediaPipe Hands - Detection Confidence: 50% - Tracking Confidence: 50% - Cooldown: 2 seconds between gestures - Max Hands: 2 simultaneous

Gesture Animation: - Three floating emojis per gesture - Upward float animation (3 seconds) - Fade-out effect - Large, visible emoji size (72pt)

Usage: 1. Enable video first (gestures require camera) 2. Click " Gestures" button 3. Perform gesture in front of camera 4. Emoji animation appears for all users 5. Gesture type logged in chat 6. Click " Stop Gestures" to disable

Performance Tips: - Good lighting improves detection - Position hands clearly in frame - Hold gesture for 0.5 seconds - Avoid rapid movements

---

5. User Interface Guide

Main Window Layout

```
[ Lan Conference App]  [ Connected]    [ ] [Server IP]
[Pass: XXXX] [Name: User] [Connect]



                                Chat & More
    Video Conference
    Area                          Chat
                                  Participants
    [User Videos]                 Files



     Screen Share             [Content Area]
    (Collapsible)



[ Start Video] [ Start Audio] [ Whiteboard]
[ Gestures] [ Share Screen] [ View Screen]
[ Leave]
```

UI Components

Header Bar

- **App Title**: Displays application name
- **Status Indicator**: Shows connection status (Connected / Disconnected)
- **Theme Toggle**: (Dark) / (Light) mode switcher
- **Server IP**: Input field for server address
- **Password**: 4-character authentication code
- **Username**: Your display name
- **Connect Button**: Initiates connection to server

Video Conference Area

- **Video Tiles**: Dynamic grid of participant video feeds
- **Screen Share Panel**: Expandable section for presentation viewing
- **Whiteboard Overlay**: Full-screen canvas for collaborative drawing

Sidebar (Chat & More)

- **Tab Navigation**: Switch between Chat, Participants, and Files
- **Group/Direct Chat Toggle**: Switch chat modes
- **Message Input**: Text field with send button
- **User List**: All connected participants
- **File Cards**: Uploaded files with download buttons

Control Bar

- **Video Control**: Start/stop camera
- **Audio Control**: Mute/unmute microphone
- **Whiteboard**: Show/hide drawing canvas
- **Gestures**: Enable/disable gesture recognition
- **Screen Share**: Start/stop presenting
- **View Screen**: Watch presenter's screen
- **Leave**: Disconnect from meeting

Theme Customization

Dark Theme (Default): - Background: #0B0B0D - Panels: #141416 - Primary: #FF6B3D (Orange accent) - Text: #F4F4F5 (Light)

Light Theme: - Background: #C8CBCD - Panels: #D6D8DA - Primary: #FF6A3D (Orange accent) - Text: #242627 (Dark)

Toggle theme with / button in header.

---

6. Security Features

Authentication System

Password Generation:

```
# Server generates on startup
charset = [A-Z, 0-9]  # 36 possible characters
length = 4
combinations = 36^4 = 1,679,616 possibilities
```

Authentication Flow:

```
Client                          Server
  |                               |
  |---hello (name, password)----->|
  |                               |
  |                     [Validate Password]
  |                               |
  |<--whiteboard_sync (success)---|  # If valid
  |   OR                          |
  |<--error (auth_failed)---------|  # If invalid
  |                               |
[Connected]                  [Log failed attempt]
```

Security Measures: - Passwords never stored permanently - Failed attempts logged with IP address - Immediate connection termination on failure - No retry mechanism (prevents brute force)

Network Security

Local Network Only: - No internet connectivity required - Data never leaves local network - No cloud storage or external servers

Protocol Security: - TCP for critical data (reliable delivery) - UDP for real-time streams (speed priority) - JSON-based messaging (human-readable, debuggable)

Best Practices: 1. Run server on trusted network only 2. Share password through secure channel (not email/SMS) 3. Restart server to generate new password regularly 4. Monitor server logs for suspicious activity

Privacy Features

Data Storage: - Video/audio streams: Not recorded (unless explicitly implemented) - Chat messages: Not persisted (memory only) - Files: Stored on server until manual deletion - Whiteboard: State stored for session only

User Control: - Explicit permissions for camera/microphone - Manual enable/disable for all features - Clear visual indicators when streaming - Leave meeting at any time

---

7. Technical Specifications

Network Configuration

Feature	Protocol	Port	Bandwidth	Latency
Control (TCP)	TCP	9000	<1 KB/s	<50ms
Video (UDP)	UDP	10000	200-400 KB/s	<100ms
Audio (UDP)	UDP	11000	30-60 KB/s	<50ms
Screen Share	TCP	9001	300-600 KB/s	<200ms
File Transfer	TCP	9002	Variable	N/A

Video Specifications

```
VIDEO_WIDTH = 320
VIDEO_HEIGHT = 240
VIDEO_FPS = 20
JPEG_QUALITY = 80
VIDEO_CHUNK = 1100 bytes
```

Bitrate Calculation:

```
Frame Size   15-30 KB (JPEG compressed)
Bitrate = 30 KB × 20 FPS = 600 KB/s = 4.8 Mbps (max)
Actual   2-3 Mbps with compression
```

Audio Specifications

```
AUDIO_RATE = 16000 Hz (16 kHz)
AUDIO_CHANNELS = 1 (Mono)
AUDIO_FORMAT = pyaudio.paInt16 (16-bit)
AUDIO_CHUNK = 256 samples
AUDIO_INPUT_CHUNK = 256 samples
```

Bitrate Calculation:

```
Bitrate = 16000 Hz × 16 bits × 1 channel = 256 kbps
Bandwidth = 256 kbps = 32 KB/s per user
```

Audio Latency Breakdown:

```
Capture: ~16ms (256 samples / 16000 Hz)
Network: <50ms (LAN)
```

```
Mixing: ~16ms (server processing)
Playback: ~16ms (256 sample buffer)
Total: ~100ms (0.1 seconds)
```

Screen Sharing Specifications

```
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 450
SCREEN_FPS = 10
SCREEN_QUALITY = 50 (JPEG)
```

Bandwidth:

```
Frame Size   30-60 KB (JPEG 50%)
Bitrate = 50 KB × 10 FPS = 500 KB/s = 4 Mbps
```

Whiteboard Protocol

Message Format:

```
{
  "type": "whiteboard_action",
  "action": "draw|shape|erase|clear|undo",
  "data": {
    "id": "uuid",
    "points": [{x, y}, ...],
    "color": "#RRGGBB",
    "width": 3,
    "timestamp": 1234567890
  }
}
```

State Synchronization: - Full state sync on client join - Incremental updates on actions - Version tracking for consistency

---

8. Troubleshooting

Connection Issues

Problem: "Connection timeout" or "Connection Failed"

Solutions: 1. Verify server is running (`python server3.py`) 2. Check IP address is correct 3. Ensure both machines on same network 4. Disable firewall temporarily to test 5. Verify no other application using ports 9000-9002, 10000-11000

Problem: "Invalid password"

Solutions: 1. Check password in server console output 2. Ensure using latest password (server generates new one on each restart) 3. Verify no extra spaces in password field 4. Password is case-sensitive (uppercase only)

Audio Problems

Problem: No audio output

Solutions: 1. Check system audio settings 2. Verify audio device not muted 3. Ensure pyaudio installed correctly 4. Test microphone in system settings 5. Check audio permissions granted to application

Problem: Echo or feedback

Solutions: 1. Use headphones instead of speakers 2. Reduce microphone sensitivity 3. Enable system echo cancellation 4. Increase distance between mic and speakers

Video Problems

Problem: Camera not detected

Solutions: 1. Close other applications using camera (Zoom, Skype, etc.) 2. Verify camera connected and powered 3. Check camera permissions in OS settings 4. Restart application 5. Try different camera if available

Problem: Choppy video

Solutions: 1. Reduce VIDEO_FPS in configuration 2. Lower JPEG_QUALITY setting 3. Check network bandwidth (use wired connection) 4. Close bandwidth-heavy applications 5. Reduce number of simultaneous video streams

Screen Sharing Issues

Problem: "Screen share denied" or cannot start

Solutions: 1. Verify no one else is currently presenting 2. Wait 2-3 seconds after previous presenter stops 3. Check screen capture permissions (macOS/Linux) 4. Ensure mss library installed correctly 5. Restart both server and client

Problem: Screen share frozen

Solutions: 1. Stop and restart screen share 2. Check network connection stability 3. Reduce SCREEN_FPS if bandwidth limited 4. Close applications running on shared screen

Whiteboard Problems

Problem: Cannot draw on whiteboard

Solutions: 1. Verify whiteboard enabled (button shows "Hide Board") 2. Check connection to server active 3. Try different drawing tool 4. Restart application if state corrupted

Problem: Drawings not synchronized

Solutions: 1. Check network latency (<200ms recommended) 2. Verify server running and processing messages 3. Restart client to resync state

Performance Optimization

For Low-End Systems:

```
# In client4.py, modify:
VIDEO_WIDTH = 160
VIDEO_HEIGHT = 120
VIDEO_FPS = 10
JPEG_QUALITY = 60
```

For High-End Systems:

```
# In client4.py, modify:
VIDEO_WIDTH = 640
VIDEO_HEIGHT = 480
VIDEO_FPS = 30
JPEG_QUALITY = 90
```

---

9. Advanced Configuration

Server Configuration

Modify server3.py:

```
# Port Configuration
TCP_PORT = 9000          # Control messages
VIDEO_UDP_PORT = 10000   # Video streams
AUDIO_UDP_PORT = 11000   # Audio streams
SCREEN_TCP_PORT = 9001   # Screen sharing
FILE_TCP_PORT = 9002     # File transfers

# Audio Buffer
AUDIO_BUFFER_SIZE = 10   # Increase for stability, decrease for latency

# Server Host
SERVER_HOST = '0.0.0.0'  # Listen on all interfaces
# or
SERVER_HOST = '192.168.1.100'  # Specific interface
```

Client Configuration

Modify client4.py:

```
# Video Settings
VIDEO_WIDTH = 320
VIDEO_HEIGHT = 240
VIDEO_FPS = 20
JPEG_QUALITY = 80

# Audio Settings
AUDIO_RATE = 16000       # Sample rate (Hz)
AUDIO_CHANNELS = 1       # Mono (1) or Stereo (2)
AUDIO_CHUNK = 256        # Samples per chunk

# Screen Sharing
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 450
SCREEN_FPS = 10
SCREEN_QUALITY = 50      # JPEG quality (1-100)
```

Custom Password

To use a fixed password instead of random:

In `server3.py`, replace:

```
SERVER_PASSWORD = generate_password()
```

With:

```
SERVER_PASSWORD = "MYPASS"  # Your custom 4-char password
```

Logging Configuration

Enable debug logging:

In both server and client:

```
logging.basicConfig(
    level=logging.DEBUG,  # Changed from INFO
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('conference.log'),  # Save to file
        logging.StreamHandler()  # Also print to console
    ]
)
```

Network Performance Tuning

For low-bandwidth networks:

```
# Reduce video quality
VIDEO_WIDTH = 160
VIDEO_HEIGHT = 120
VIDEO_FPS = 10
JPEG_QUALITY = 50

# Reduce screen share quality
SCREEN_WIDTH = 640
SCREEN_HEIGHT = 360
SCREEN_FPS = 5
SCREEN_QUALITY = 30
```

For high-bandwidth networks:

```
# Increase video quality
VIDEO_WIDTH = 640
VIDEO_HEIGHT = 480
VIDEO_FPS = 30
JPEG_QUALITY = 95

# Increase screen share quality
SCREEN_WIDTH = 1280
SCREEN_HEIGHT = 720
SCREEN_FPS = 15
SCREEN_QUALITY = 70
```

File Storage Location

Change server file storage:

In `server3.py`:

```
# Default
os.makedirs("server_files", exist_ok=True)

# Custom location
FILE_STORAGE = "/path/to/shared/storage"
os.makedirs(FILE_STORAGE, exist_ok=True)
```

Change client download location:

In `client4.py`:

```
# Default
DOWNLOADS_DIR = os.path.join(os.path.expanduser("~"), "Downloads", "ConferenceFiles")

# Custom location
DOWNLOADS_DIR = "/custom/download/path"
os.makedirs(DOWNLOADS_DIR, exist_ok=True)
```

Running as System Service

Linux (systemd):

Create `/etc/systemd/system/Lan-server.service`:

```
[Unit]
Description=Lan Conference Server
After=network.target

[Service]
Type=simple
User=youruser
WorkingDirectory=/path/to/Lan
ExecStart=/usr/bin/python3 /path/to/Lan/server.py
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Enable and start:

```
sudo systemctl enable Lan-server
sudo systemctl start Lan-server
sudo systemctl status Lan-server
```

Command-Line Arguments

Add argument parsing to server3.py:

```
import argparse

parser = argparse.ArgumentParser(description='Lan Conference Server')
parser.add_argument('--password', type=str, help='Set custom password')
parser.add_argument('--port', type=int, default=9000, help='TCP control port')
args = parser.parse_args()

if args.password:
    SERVER_PASSWORD = args.password
else:
    SERVER_PASSWORD = generate_password()

TCP_PORT = args.port
```

Run with:

```
python server3.py --password ABCD --port 9000
```

Appendix: Quick Reference

Server Commands

```
# Start server
python server3.py

# Start with custom password
python server3.py --password MYPASS

# View logs
tail -f conference.log
```

Client Shortcuts

- **Enter**: Send chat message
- **Ctrl+Q**: Quit application
- **Theme toggle**: Switch dark/light mode

Default Ports

- 9000: TCP Control
- 10000: UDP Video
- 11000: UDP Audio
- 9001: TCP Screen
- 9002: TCP Files

File Locations

- Server files: `./server_files/`
- Client downloads: `~/Downloads/ConferenceFiles/`
- Logs: `./conference.log` (if enabled)