High-Fidelity Wireless Audio Transport in Isolated Network Environments: A Comparative Analysis of Open-Source Android-to-Linux Streaming Architectures

1. Introduction and Architectural Overview

The convergence of high-resolution mobile audio playback and embedded Linux systems presents a distinct set of engineering challenges, particularly when bypassing proprietary, lossy protocols like Bluetooth in favor of bit-perfect transmission. As mobile devices increasingly serve as the primary interface for high-fidelity streaming services (Tidal, Qobuz, Apple Music), the necessity for a robust, lossless transport mechanism to dedicated audio hardware has become paramount. This report provides a comprehensive architectural analysis and implementation guide for establishing a lossless, high-fidelity audio stream from an Android source to a headless Raspberry Pi receiver over WiFi.

The core objective is to replicate the seamless functionality of Apple's AirPlay within an open ecosystem, utilizing open-source protocols to ensure bit-perfect audio transmission.[1] A critical constraint addressed herein is the "split-tunneling" or "dual-homing" network topology: maintaining a local WiFi connection for high-bandwidth audio transport to the Raspberry Pi while simultaneously preserving the Android device's cellular connection for internet access. This requirement necessitates a departure from standard consumer networking configurations, requiring precise manipulation of the Dynamic Host Configuration Protocol (DHCP) and the Android Connectivity Manager.[2]

The analysis identifies three primary protocol stacks suitable for this application: **AirPlay emulation** (via shairport-sync and AirMusic), **DLNA/UPnP** (via gmrender-resurrect and BubbleUPnP), and **Real-time RTP streaming** (via Roc Toolkit or Snapcast). Each architecture offers distinct advantages regarding latency, jitter resilience, and integration complexity. The report concludes that while AirPlay emulation offers the highest user convenience, the Roc Toolkit provides superior packet loss resilience for volatile WiFi environments via Forward Erasure Correction (FEC).[4]

1.1 The Limitations of Bluetooth vs. WiFi Transport

The user requirement explicitly excludes Bluetooth, a decision supported by rigorous technical necessity for lossless transmission. Bluetooth audio transmission traditionally relies

on the A2DP (Advanced Audio Distribution Profile), which necessitates transcoding source audio into lossy codecs such as SBC, AAC, or aptX to fit within the limited bandwidth of the Bluetooth radio (typically < 1 Mbps effective throughput for audio).[6] Even high-bandwidth proprietary variants like LDAC, which claims 990 kbps throughput, utilize lossy compression algorithms that discard psychoacoustically "irrelevant" data to maintain stream stability. In congested 2.4 GHz environments, Bluetooth bitpools aggressively throttle, reducing bitrate and dynamic range, effectively preventing true 16-bit/44.1kHz (Red Book) lossless transmission.[6]

In contrast, WiFi (802.11n/ac) offers theoretical throughputs ranging from 150 Mbps to over 1 Gbps. A stereo CD-quality stream (16-bit, 44.1 kHz PCM) requires a mere 1.411 Mbps of raw throughput. Even High-Resolution Audio (24-bit, 192 kHz) demands only ~9.2 Mbps. Consequently, WiFi provides ample headroom for uncompressed Pulse-Code Modulation (PCM) or lossless compression (FLAC/ALAC) without psychoacoustic discarding, rendering it the only viable transport for true high-fidelity wireless audio where bit-perfect integrity is required.[1]

1.2 The Android Audio Subsystem Challenge

Replicating "AirPlay-like" system-wide audio mirroring on Android is historically complex due to the operating system's security architecture. Prior to Android 10, applications were sandboxed, preventing one app from capturing the audio output of another without "root" privileges (superadmin access).[9] This limitation forced users to rely on specific media players with built-in casting capabilities rather than system-wide mirroring.

With the introduction of Android 10 (API level 29), Google introduced the AudioPlaybackCapture API. This API allows apps to capture audio from other apps, provided the target app allows it (via allowAudioPlaybackCapture="true" in its manifest) and the capturing app holds the RECORD_AUDIO permission.[10] This evolution enables non-root solutions like **AirMusic** and **BubbleUPnP** to capture system audio and redirect it to network protocols.[12] However, Digital Rights Management (DRM) restrictions remain; video streaming apps often block audio capture to prevent piracy, though music services like Spotify, Tidal, and YouTube Music generally permit it.[14]

1.3 Linux Audio Architecture: ALSA, PulseAudio, and PipeWire

The receiver side—the Raspberry Pi—runs a Linux stack. The choice of audio subsystem is critical for bit-perfect playback. The architecture typically consists of layers:

- **ALSA (Advanced Linux Sound Architecture):** The kernel-level driver interface. It provides the lowest latency and direct hardware access (DHA). For a dedicated "audiophile" endpoint, outputting directly to ALSA is preferred to avoid the resampling that user-space sound servers might introduce.[15]
- **PulseAudio:** A sound server that sits atop ALSA. While powerful for desktop mixing, it

historically introduced latency and forced resampling (e.g., converting 44.1 kHz sources to 48 kHz default), which violates the lossless/bit-perfect requirement unless meticulously configured.[17]
- **PipeWire:** The modern replacement for PulseAudio, offering low-latency processing suitable for professional audio. It is increasingly relevant for protocols like Roc Toolkit, which integrates natively with PipeWire to manage graph-based audio routing.[5]

For a headless, dedicated receiver, this report prioritizes direct **ALSA** interaction or minimal **PipeWire** configurations to minimize jitter, latency, and resampling artifacts.

2. Network Topology and Routing Engineering

The user's requirement to use cellular data for the internet while connected to the Pi's WiFi hotspot presents a classic "split-horizon" or "dual-homing" routing problem. By default, Android (and most operating systems) assumes that a WiFi connection implies Internet access and will update the routing table to send all traffic (0.0.0.0/0) through the WiFi interface (wlan0). If the Pi has no internet upstream, the phone loses connectivity, or the OS disconnects from the WiFi to preserve user experience.[2]

2.1 The "No Internet" Hotspot Configuration

To force Android to use Cellular for WAN (Wide Area Network) traffic and WiFi only for LAN (Local Area Network) traffic, we must manipulate the DHCP handshake. The Dynamic Host Configuration Protocol (DHCP) provides a client with an IP address, Subnet Mask, DNS Server, and **Default Gateway** (Router). The Default Gateway (Option 3) instructs the client: "Send all traffic destined for unknown networks to this IP."

If the Raspberry Pi's DHCP server (dnsmasq) advertises itself as the gateway, Android will route internet requests to the Pi, where they will terminate. The solution is to configure the Pi to **not** advertise a default gateway. When Android receives an IP address but no gateway, it classifies the network as "Local Only" and keeps the cellular data connection active for internet traffic.[20]

2.2 Deep Dive: Dnsmasq Configuration

dnsmasq is the lightweight DHCP and DNS server typically used by hostapd (Host Access Point Daemon) on the Raspberry Pi. Standard configuration often looks like this:

Bash

```
dhcp-option=3,192.168.4.1  # Advertises Pi as Gateway
```

To disable the gateway advertisement, specific syntax is required. The documentation and user experiences [21] indicate that sending an empty Option 3 is the correct approach to prevent the client from installing a default route via the WiFi interface.

Correct Syntax for Suppression:

Bash

```
dhcp-option=3
```

Or, more explicitly to ensure no routing is pushed:

Bash

```
dhcp-option=option:router
```

By simply omitting the IP address after the option number, dnsmasq suppresses the transmission of that option. Additionally, Option 6 (DNS) should also be suppressed or set to a null value to prevent Android from trying to resolve domain names via the offline Pi, which would result in DNS timeouts even if the route exists via cellular.[22]

Table 1: DHCP Option Configuration for Split-Tunneling

DHCP Option	Standard Value	Split-Tunnel Value	Effect on Android Client
Option 3 (Router)	192.168.4.1	dhcp-option=3 (Empty)	Prevents Android from setting WiFi as default gateway. Forces WAN traffic to Cellular.
Option 6 (DNS)	192.168.4.1 or 8.8.8.8	dhcp-option=6 (Empty)	Prevents DNS lookups over WiFi. Forces DNS

			resolution via Cellular.
Option 1 (Subnet)	255.255.255.0	255.255.255.0	Defines local subnet scope.

2.3 Android Client Behavior and Static IP Fallback

While the server-side DHCP manipulation is the most elegant solution, Android implementations vary by vendor (e.g., Samsung OneUI vs. Pixel UI). Some aggressive implementations may disconnect from a WiFi network if it fails a "connectivity check" (pinging Google servers).

If DHCP manipulation fails, manual static IP configuration on the phone is the fallback mechanism.[2]

- **IP:** 192.168.4.2 (in the Pi's subnet)
- **Gateway:** *Leave Empty* (or 0.0.0.0 if forced by UI)
- **DNS:** *Leave Empty* (or 8.8.8.8, though this will be routed via cellular)

Research indicates that Android 11+ has deprecated some static IP settings in the UI for WiFi, making the server-side DHCP suppression via dnsmasq the more robust and user-friendly approach for modern devices.[25] Furthermore, enabling "Mobile Data Always Active" in Android Developer Options can assist in maintaining the cellular link during the handover, although the "No Gateway" DHCP setting usually negates the need for this developer toggle.[2]

3. Protocol Option A: AirPlay Emulation (Shairport-Sync)

This approach most closely matches the user's request ("similar to how AirPlay works"). It utilizes the Apple RAOP (Remote Audio Output Protocol), which is renowned for its high-fidelity ALAC (Apple Lossless Audio Codec) transport and metadata capabilities.

3.1 The Sender: Android Implementation via AirMusic

Since Android does not support AirPlay natively at the system level, a bridge application is required. **AirMusic** (formerly AirAudio) and **Cast Receiver** are the primary options.[14] AirMusic is particularly notable for its integration of the AudioPlaybackCapture API.

- **Mechanism:** On Android 10+, AirMusic uses the native capture API to record the PCM output of eligible apps. It then encodes this stream into ALAC and wraps it in the AirPlay protocol (RTSP/RTP).

- **Root vs. Non-Root:**
  - **Non-Root (Android 10+):** Captures audio from apps that allow it (Spotify, YouTube Music, Deezer). Some apps (Apple Music, Netflix) may block capture via DRM flags.[12]
  - **Rooted:** AirMusic can inject a system-level driver to capture *all* audio, bypassing app-specific restrictions.
- **Latency Management:** AirPlay 1 uses a UDP transport with retransmissions, while AirPlay 2 introduces larger buffers. AirMusic creates a virtual AirPlay sender that discovers the Raspberry Pi via mDNS (Bonjour/Avahi).[12]

3.2 The Receiver: Shairport-Sync

shairport-sync is a mature, widely adopted open-source AirPlay receiver for Linux. It is distinct from the older shairport in its focus on audio synchronization (hence the name), making it ideal for multi-room setups, though perfectly capable of single-room high-fidelity playback.[30]

- **Audio Quality:** AirPlay streams using the **ALAC** format at 44.1 kHz / 16-bit (CD Quality). shairport-sync decodes this bit-perfectly and passes it to the ALSA backend.[32]
- **Interpolation and Drift:** A critical feature of shairport-sync is its ability to synchronize the stream to the DAC's clock. Audio sources and sinks run on independent crystals; over time, clock drift causes buffer under/overruns. shairport-sync uses libsoxr (SoX Resampler) to perform high-quality interpolation, subtly adjusting the playback rate to match the source without audible pitch shifting or dropouts. This is superior to basic "pipe" streaming which simply drops frames when buffers fill.[32]
- **Metadata:** It supports sending metadata (Artist, Track, Album Art) via a pipe or stdout, which can be used to drive a display on the headless Pi if desired.[31]

4. Protocol Option B: UPnP/DLNA (The Universal Standard)

Universal Plug and Play (UPnP) and the Digital Living Network Alliance (DLNA) standards are the traditional methods for media streaming on local networks. This is a robust fallback if AirPlay emulation encounters specific compatibility issues.

4.1 The Sender: BubbleUPnP

BubbleUPnP is the industry standard for DLNA on Android. It acts as a Control Point, Media Server, and Renderer.[34]

- **Audio Cast:** This feature records system audio (similar to AirMusic) and wraps it in a WAV/LPCM stream to send to a DLNA renderer. This effectively creates a "live stream" over HTTP/DLNA.[36]
- **Transcoding:** If the receiver doesn't support the source format, BubbleUPnP can transcode on the fly (e.g., converting incompatible bit depths), ensuring playback

compatibility.[34]

4.2 The Receiver: Gmrender-Resurrect

gmrender-resurrect is a resource-efficient, headless UPnP renderer based on the GStreamer multimedia framework.[38]

- **Function:** It advertises itself on the network as a render endpoint via SSDP (Simple Service Discovery Protocol). When it receives a stream (like the PCM stream from BubbleUPnP), GStreamer pipelines handle the decoding and output to ALSA.
- **Format Support:** Because it uses GStreamer, it supports virtually any codec (FLAC, WAV, MP3, AAC, OGG) installed on the Pi, making it highly versatile.[39]
- **Configuration:** It is typically run as a systemd service, utilizing a Universally Unique Identifier (UUID) to persist its identity across reboots.[39]

Verdict: While versatile, DLNA is based on TCP/HTTP and can be "chattier" than UDP-based protocols. The latency for "live" casting via DLNA is often higher (3-5 seconds) compared to AirPlay or RTP solutions, making it less suitable for video synchronization but excellent for pure music.

5. Protocol Option C: Real-Time RTP Streaming (Roc Toolkit)

For a user prioritizing "open-source projects" and technical robustness over ecosystem emulation, **Roc Toolkit** is the superior, modern engineering choice. It is designed specifically for real-time streaming over unreliable networks like WiFi.[4]

5.1 The Protocol: RTP + FEC

Unlike TCP-based protocols (HTTP/DLNA) that request retransmission of lost packets (causing latency spikes and buffer halts), Roc uses RTP (Real-time Transport Protocol) over UDP combined with **Forward Erasure Correction (FEC)**.[4]

- **FEC Explained:** The sender transmits redundant parity packets (e.g., Reed-Solomon codes). If a packet is lost due to WiFi interference (common in 2.4GHz bands), the receiver can reconstruct the missing data mathematically from the parity packets without asking for a retransmission. This guarantees continuous playback with fixed, low latency, essential for audiophile listening where dropouts are unacceptable.

5.2 The Sender: Roc Droid

The Roc Droid application (available on F-Droid) implements the Roc sender for Android.[40]

- **Function:** It captures audio (via mic or internal system audio if supported/rooted) and streams it to a target IP.

- **Configuration:** Users define the FEC encoding parameters (e.g., Reed-Solomon m=8) and target latency. This allows fine-tuning the trade-off between bandwidth usage and reliability.

5.3 The Receiver: Roc Toolkit on Linux

On the Raspberry Pi, the Roc receiver (roc-recv) listens on a specific port.

- **PipeWire Integration:** Roc integrates natively with PipeWire, which is becoming the standard Linux audio server. This allows for complex routing (e.g., applying equalization or room correction on the Pi before outputting to the DAC).[5]

Verdict: Roc Toolkit offers the best resilience to WiFi interference and the lowest latency. However, it requires more manual configuration (IP targets, port management) and lacks the "auto-discovery" metadata polish of AirPlay/DLNA.

6. Implementation Guide: The "AirPi" Headless Receiver

Based on the synthesis of user requirements (seamless, high-fidelity, open-source, non-Bluetooth), the recommended implementation utilizes **AirPlay emulation** via shairport-sync as the primary protocol due to its balance of quality (ALAC), ease of use (via AirMusic), and metadata support. The network layer will be secured via dnsmasq.

6.1 Hardware Prerequisites

- **Raspberry Pi:** Model 3B+ or 4B recommended for stable WiFi throughput, though Zero 2 W is sufficient for audio-only workloads.
- **SD Card:** 8GB+ with Raspberry Pi OS Lite (64-bit preferred for Pi 3/4).[8]
- **DAC (Digital-to-Analog Converter):** The Pi's onboard 3.5mm jack uses PWM (Pulse Width Modulation) and is low fidelity. An I2S DAC HAT (e.g., HiFiBerry, Pimoroni) or a high-quality USB DAC is mandatory for "lossless" listening.[1]

6.2 Phase 1: Operating System and Network Configuration

Step 1: Headless OS Setup
Flash Raspberry Pi OS Lite. Before ejecting the SD card, edit the config.txt in the boot partition to enable the overlay for your specific DAC (e.g., dtoverlay=hifiberry-dac). Create an empty file named ssh in the boot partition to enable remote access.44
Step 2: WiFi Access Point Setup (hostapd)
Install the necessary packages:

```
Bash
```

```
sudo apt update
sudo apt install hostapd dnsmasq
```

Configure /etc/hostapd/hostapd.conf to create the hotspot. Using 5GHz is highly recommended for audio bandwidth stability.

- interface=wlan0
- hw_mode=a (Enforces 5GHz)
- channel=36 (Select a non-DFS channel)
- wmm_enabled=1 (WiFi Multimedia QoS, essential for prioritizing streaming traffic).[46]

Step 3: The "No Internet" DHCP Configuration (dnsmasq)
This is the critical step to satisfy the user's split-tunneling requirement.
Edit /etc/dnsmasq.conf:

Bash

```
interface=wlan0
bind-interfaces
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
# Vital: Suppress Default Gateway (Option 3) and DNS (Option 6)
dhcp-option=3
dhcp-option=6
```

Note: Depending on the specific dnsmasq version, the syntax dhcp-option=3 (with no value) or dhcp-option=option:router ensures that the Pi does not advertise itself as a gateway.[21]

Step 4: Static IP for wlan0
Configure dhcpcd.conf to assign a static IP to the Pi's wireless interface so it acts as the stable anchor:

Bash

```
interface wlan0
static ip_address=192.168.4.1/24
nohook wpa_supplicant
```

6.3 Phase 2: Audio Receiver Installation

Step 1: ALSA Configuration
List available audio devices with aplay -l.
Disable the onboard audio in /boot/config.txt (dtparam=audio=off) to ensure the DAC HAT consistently appears as hw:0 or card 0.
Step 2: Shairport-Sync (AirPlay Receiver)
To utilize modern features like soxr resampling and metadata, build from source rather than using the potentially outdated apt repository packages.31

Bash

```
# Install Build Dependencies
sudo apt install autoconf libtool libdaemon-dev libasound2-dev libpopt-dev libconfig-dev avahi-daemon libavahi-client-dev libssl-dev libsoxr-dev

# Clone and Build
git clone https://github.com/mikebrady/shairport-sync.git
cd shairport-sync
autoreconf -i -f
./configure --sysconfdir=/etc --with-alsa --with-avahi --with-ssl=openssl --with-systemd --with-metadata --with-soxr
make
sudo make install
```

Step 3: Configuring Shairport-Sync
Edit /etc/shairport-sync.conf to optimize for the HAT:

Code snippet

```
general = {
  name = "AirPi HighFi";
  interpolation = "soxr"; // High-quality resampling
};

alsa = {
  output_device = "hw:0"; // Direct hardware access
  mixer_control_name = "Digital"; // Use hardware mixer if available
};
```

Enable the service: sudo systemctl enable shairport-sync.

Step 4: Gmrender-Resurrect (DLNA Fallback)
For redundancy, install gmrender-resurrect:

```
Bash
```

```
sudo apt install gstreamer1.0-alsa gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly
git clone https://github.com/hzeller/gmrender-resurrect.git
# Build and install steps mirroring the standard autotools process
```

Create a systemd service file that runs it with a persistent UUID (uuidgen) to ensure Android clients recognize it as the same device after reboots.[39]

6.4 Phase 3: Android Client Configuration

1. **WiFi Connection:** Connect the phone to the "AirPi" hotspot.
2. **Verification:** Android should display a notification: "This network has no internet access. Stay connected?" Select **Yes** (or "Don't ask again"). The LTE/5G icon should remain active in the status bar alongside the WiFi icon (often with a small 'x' or exclamation mark indicating no WAN).
3. **Streaming App:**
    - **AirMusic:** Launch the app. Grant "Screen Casting/Recording" permissions. Select "AirPlay" and tap on "AirPi HighFi."
        - *Result:* All system audio (Spotify, YouTube, Local Files) is captured, encoded to ALAC, and sent to the Pi via WiFi.
    - **BubbleUPnP:** Open the app. Select "Audio Cast" from the menu. Select the "AirPi" (DLNA renderer). Audio begins playing via DLNA LPCM.

7. Performance Nuances and Troubleshooting

7.1 Latency vs. Synchronization

AirPlay inherently introduces a 2-second buffer to ensure synchronization and stability. This is acceptable for music but problematic for gaming or video where lip-sync is required. shairport-sync allows tweaking this via the audio_backend_latency_offset_in_seconds parameter in the config file, though lowering it reduces interference immunity.[32] For video sync, the AirMusic app on Android offers a "Audio-Delay" slider to manually compensate for the transport latency.

7.2 WiFi Congestion and Jitter

Streaming 16/44.1 PCM requires stable throughput. Jitter—the variance in packet arrival time—is the enemy of smooth playback.

- **Frequency:** Ensure the Pi is using 5 GHz WiFi (requires Pi 3B+ or newer). The 2.4 GHz spectrum is often too crowded with Bluetooth and microwave interference for reliable real-time streaming.
- **QoS:** Check hostapd settings for ieee80211n=1 (Enable 802.11n) and wmm_enabled=1 (QoS).
- **FEC:** If using Roc Toolkit, increase the FEC blocks in the sender configuration to recover lost packets at the cost of a slight bandwidth increase (e.g., 20-30% overhead).

7.3 Clock Drift and Interpolation

Because the Android phone and Raspberry Pi have independent crystal oscillators, clock drift will occur over long listening sessions. If the sender sends 44,100 samples per second but the receiver plays 44,101 samples per second, the buffer will eventually empty (underrun). shairport-sync handles this actively by monitoring the fill level of the ALSA buffer and using libsoxr to make micro-adjustments (interpolation) to the audio stream. This keeps the stream synchronized without audible artifacts, a significant advantage over "dumb" pipe solutions like netcat or basic DLNA renderers.16

8. Conclusion

The analysis confirms that a high-fidelity, open-source audio streaming solution matching the user's stringent requirements is entirely feasible. By combining **Shairport-Sync** on a headless Raspberry Pi with the **AirMusic** application on Android, one achieves an "AirPlay-grade" experience with lossless ALAC transport. The critical network constraint—simultaneous Cellular Internet and WiFi Audio—is elegantly solved not by complex routing tables on the phone, but by the strategic suppression of DHCP Option 3 within the Raspberry Pi's dnsmasq configuration. This architecture provides a robust, audiophile-grade transport independent of proprietary cloud ecosystems or lossy Bluetooth compression. The inclusion of Roc Toolkit as an alternative path offers a solution for environments requiring extreme low latency or resilience to packet loss, ensuring the system can adapt to various listening conditions.

9. Appendix: Bandwidth Calculations

To assist in network planning, the following table details the bandwidth requirements for various lossless formats supported by this architecture.

Table 2: Network Bandwidth Requirements for Lossless Audio

Format	Resolution	Bitrate	Approx. WiFi Throughput

		(Uncompressed)	Required
CD Quality	16-bit / 44.1 kHz	1.411 Mbps	~2.0 Mbps
DVD Quality	24-bit / 48 kHz	2.304 Mbps	~3.5 Mbps
High-Res	24-bit / 96 kHz	4.608 Mbps	~6.0 Mbps
Studio Master	24-bit / 192 kHz	9.216 Mbps	~12.0 Mbps

Note: "WiFi Throughput Required" includes overhead for protocol headers (RTP/TCP/IP), encryption, and interference headroom. Even the highest resolution is well within the capability of 802.11n (150 Mbps) or 802.11ac (433+ Mbps).

Works cited

1. Raspberry Pi As Low-cost Audio Streaming Box - Instructables, accessed December 29, 2025, https://www.instructables.com/Raspberry-Pi-as-low-cost-audio-streaming-box/
2. Using Mobile Internet/Data While Connected To A WiFi Network ..., accessed December 29, 2025, https://docs.safesky.app/books/safesky-pilot-playbook/page/using-mobile-internetdata-while-connected-to-a-wifi-network-iosandroid
3. Enable usage of mobile data while connected to a WiFi without ..., accessed December 29, 2025, https://android.stackexchange.com/questions/231538/enable-usage-of-mobile-data-while-connected-to-a-wifi-without-internet-connectio
4. Features — Roc Toolkit 0.4.0, accessed December 29, 2025, https://roc-streaming.org/toolkit/docs/about_project/features.html
5. Roc Toolkit 0.4 + updated tutorial for live audio streaming, accessed December 29, 2025, https://gavv.net/articles/roc-0.4/
6. Send music from cellphone App to Rpi audio server (no bluetooth), accessed December 29, 2025, https://forums.raspberrypi.com/viewtopic.php?t=346505
7. Bluetooth-Audio -> SnapCast - General - Mopidy Discourse, accessed December 29, 2025, https://discourse.mopidy.com/t/bluetooth-audio-snapcast/5519
8. Raspberry Pi Streaming: My Guide to what it is, why you want one ..., accessed December 29, 2025, https://www.reddit.com/r/BudgetAudiophile/comments/jeax1g/raspberry_pi_streaming_my_guide_to_what_it_is_why/
9. How to stream media (specifically audio) from an Android device to ..., accessed December 29, 2025, https://learn.microsoft.com/en-us/answers/questions/4369508/how-to-stream-m

edia-(specifically-audio)-from-an-a
10. System audio streaming on Android over Webrtc - Stack Overflow, accessed December 29, 2025, https://stackoverflow.com/questions/68576071/system-audio-streaming-on-android-over-webrtc
11. snapcast server on android · Issue #271 - GitHub, accessed December 29, 2025, https://github.com/badaix/snapcast/issues/271
12. AirMusic lets you use Apple's AirPlay on Android without root - Reddit, accessed December 29, 2025, https://www.reddit.com/r/Android/comments/krq85e/airmusic_lets_you_use_apples_airplay_on_android/
13. Add Airplay to your Android Phone - News - Arylic Forum, accessed December 29, 2025, https://forum.arylic.com/t/add-airplay-to-your-android-phone/180
14. AirMusic - stream your music!, accessed December 29, 2025, https://www.airmusic.app/
15. I finally found a way to stream audio from OSMC (RPi) to your ..., accessed December 29, 2025, https://discourse.osmc.tv/t/i-finally-found-a-way-to-stream-audio-from-osmc-rpi-to-your-smartphone-android-over-wifi-lan/17998
16. snapcast/snapcast: Synchronous multiroom audio player - GitHub, accessed December 29, 2025, https://github.com/snapcast/snapcast
17. ansible-role-audio/notes.md at master - GitHub, accessed December 29, 2025, https://github.com/crowding/ansible-role-audio/blob/master/notes.md
18. Snapcast – Synchronous multi-room audio player | Hacker News, accessed December 29, 2025, https://news.ycombinator.com/item?id=21905120
19. Android 12 - how to use mobile data while connected to wifi with no ..., accessed December 29, 2025, https://www.reddit.com/r/oneplus/comments/tq5tp7/android_12_how_to_use_mobile_data_while_connected/
20. [Dnsmasq-discuss] DHCP with no default route and no DNS, accessed December 29, 2025, https://lists.thekelleys.org.uk/pipermail/dnsmasq-discuss/2005q2/000256.html
21. I don't want my DHCP to be a default gateway - Super User, accessed December 29, 2025, https://superuser.com/questions/306121/i-dont-want-my-dhcp-to-be-a-default-gateway
22. How to prevent dnsmasq to send DNS and default gateway DHCP ..., accessed December 29, 2025, https://forum.archive.openwrt.org/viewtopic.php?id=35084
23. Dnsmasq + DHCP: how to return empty DNS to all clients, accessed December 29, 2025, https://stackoverflow.com/questions/37996123/dnsmasq-dhcp-how-to-return-empty-dns-to-all-clients
24. How to Set Up a Static IP Address | PCMag, accessed December 29, 2025, https://www.pcmag.com/how-to/how-to-set-up-a-static-ip-address
25. Configuring the Device to Use a Static IP Address, accessed December 29, 2025,

https://docs.zebra.com/us/en/mobile-computers/handheld/ps2-series/ps20-product-reference-guide/ps20-product-reference-guide/configuring-the-device-to-use-a-static-ip-addres-0.html
26. Setting Up a Static IP for Wi-Fi - Esper Help, accessed December 29, 2025, https://help.esper.io/hc/en-us/articles/14013304846993-Setting-Up-a-Static-IP-for-Wi-Fi
27. can't use mobile data when connected to wifi without internet ..., accessed December 29, 2025, https://eu.community.samsung.com/t5/other-galaxy-s-series/can-t-use-mobile-data-when-connected-to-wifi-without-internet/td-p/3490434
28. Cast Receiver - DLNA & AirPlay - Apps on Google Play, accessed December 29, 2025, https://play.google.com/store/apps/details?id=com.aicareles.dlnaserver
29. AirMusic - stream your music! - Apps on Google Play, accessed December 29, 2025, https://play.google.com/store/apps/details?id=app.airmusic.pro
30. Raspberry PI Music Server – The Complete Guide (2025 Edition), accessed December 29, 2025, https://www.musicservertips.com/setup-guides/raspberry-pi-music-server-guide/
31. Multi-Room Audio: Snapcast, Spotify Connect, Airplay, and Home ..., accessed December 29, 2025, https://www.reddit.com/r/homeassistant/comments/y1gu3f/multiroom_audio_snapcast_spotify_connect_airplay/
32. shairport-sync — Debian unstable, accessed December 29, 2025, https://manpages.debian.org/unstable/shairport-sync/shairport-sync.7.en.html
33. Shairport Metadata - Raspberry Pi Forums, accessed December 29, 2025, https://forums.raspberrypi.com/viewtopic.php?t=93928
34. Features and Requirements | BubbleUPnP Server, accessed December 29, 2025, https://www.bubblesoftapps.com/bubbleupnpserver2/docs/features_and_requirements.html
35. BubbleUPnP for DLNA/Chromecast - Apps on Google Play, accessed December 29, 2025, https://play.google.com/store/apps/details?id=com.bubblesoft.android.bubbleupnp
36. Bubble UPnP Audio cast, accessed December 29, 2025, https://community.naimaudio.com/t/bubble-upnp-audio-cast/22345
37. [App] BubbleUPnP adds "Audio Cast", with the ability to capture and ..., accessed December 29, 2025, https://www.reddit.com/r/Android/comments/1x7qko/app_bubbleupnp_adds_audio_cast_with_the_ability/
38. Stream all audio FROM my android phone TO pulseaudio, accessed December 29, 2025, https://askubuntu.com/questions/963992/stream-all-audio-from-my-android-phone-to-pulseaudio
39. gmrender-resurrect/INSTALL.md at master - GitHub, accessed December 29, 2025, https://github.com/hzeller/gmrender-resurrect/blob/master/INSTALL.md

40. Applications — Roc Toolkit 0.4.0, accessed December 29, 2025, https://roc-streaming.org/toolkit/docs/tools/applications.html
41. roc-streaming/roc-droid: Roc for Android! - GitHub, accessed December 29, 2025, https://github.com/roc-streaming/roc-droid
42. Roc Droid - IzzyOnDroid F-Droid Repository, accessed December 29, 2025, https://apt.izzysoft.de/fdroid/index/apk/org.rocstreaming.rocdroid
43. Echo Cancellation on Linux with PipeWire + ROC - Gist - GitHub, accessed December 29, 2025, https://gist.github.com/fathonix/05de5398f3f92ccd51006a91a819bbd7
44. Multi-room audio with Snapcast and Raspberry Pi - oyvn, accessed December 29, 2025, https://oyvn.no/multi-room-audio-with-snapcast
45. Setting up a headless Raspberry Pi the hard way | by Dragan Čečavac, accessed December 29, 2025, https://medium.com/@celecavac/setting-up-a-headless-raspberry-pi-the-hard-way-15e78e644d50
46. Rpi3 Access Point/Hotspot Connects but no Internet, accessed December 29, 2025, https://raspberrypi.stackexchange.com/questions/53483/rpi3-access-point-hotspot-connects-but-no-internet
47. How to setup Raspberry Pi as Access Point Router (AP + Hotspot) -, accessed December 29, 2025, https://peakup.org/blog/how-to-setup-raspberry-pi-as-access-point-router-ap-hotspot/
48. Pi as DLNA / UPnP renderer (headless) : r/raspberry_pi - Reddit, accessed December 29, 2025, https://www.reddit.com/r/raspberry_pi/comments/25esc7/pi_as_dlna_upnp_renderer_headless/