FTP Phone Connection - Technical Research Report

Problem Statement

Unable to automatically detect and connect to phone's FTP server across multiple network configurations (phone hotspot, PC hotspot, router network).

Technical Context

Device Information

- **Phone MAC Address**: 64-dd-e9-5c-e3-f3
- **FTP Server Port**: 2121
- **FTP Credentials**: Username: 14ag, Password: [configured]
- **Phone Model**: [TO BE FILLED]
- **FTP App Used**: [TO BE FILLED]

Network Scenarios

1. **Phone as Hotspot**: Device acts as gateway, typically 192.168.43.1 or 192.168.49.1
2. **PC as Hotspot**: Windows Mobile Hotspot, network 192.168.137.x
3. **Router Network**: Both devices on same WiFi, network 192.168.100.x

Current Implementation Challenges

Challenge 1: MAC Address Detection

- **Issue**: Windows ARP cache doesn't always contain the phone's MAC address
- **Current approach**: Ping sweep followed by ARP lookup
- **Limitation**: ARP entries expire quickly (2-10 minutes)
- **Research needed**: Methods to force ARP cache population or alternative device identification

Challenge 2: Network Identification

- **Issue**: Determining which network configuration is active
- **Current approach**: Sequential checking (gateway → PC hotspot → router)
- **Limitation**: No reliable way to detect if PC hotspot is active via command line
- **Research needed**: WMI queries or registry keys that indicate hotspot status

Challenge 3: Port Scanning

- **Issue**: Testing FTP port (2121) availability without specialized tools

- **Current approach**: PowerShell TcpClient or telnet
- **Limitation**: Windows Firewall may block attempts
- **Research needed**: Native Windows methods for port checking

Specific Technical Questions

1. Enhanced MAC Resolution

Question: What Windows API or WMI query can reliably map IP addresses to MAC addresses without relying on ARP cache?

Context: Need method that works across:

- Different network types (WiFi, Ethernet, Hotspot)
- Without administrative privileges
- In real-time (not cached data)

2. Mobile Hotspot Detection

Question: How can we programmatically detect if Windows 10/11 Mobile Hotspot is active and get connected clients?

Possible approaches to research:

- `netsh wlan show hostednetwork` (legacy, doesn't work for Win10 Mobile Hotspot)
- WMI class `Win32_NetworkAdapter` or `MSFT_NetAdapter`
- Registry: `HKLM\SYSTEM\CurrentControlSet\Services\icssvc`
- PowerShell: `Get-NetAdapter` with specific properties

3. DHCP Lease Information

Question: Can we query DHCP server (when PC is hotspot) to get client list with MAC addresses?

Research areas:

- `netsh dhcp server` commands
- WMI DHCP classes
- Internet Connection Sharing (ICS) API

4. NetBIOS/LLMNR Resolution

Question: Can we use NetBIOS name resolution or LLMNR to identify the phone without knowing its IP?

Implementation ideas:

- Register phone with consistent NetBIOS name

- Use `nbtstat` or LLMNR queries
- mDNS/Bonjour protocol implementation

Code Snippets Needing Optimization

Current ARP Lookup (Inefficient)

```batch
for /l %%i in (1,1,254) do (
    ping -n 1 -w 200 192.168.x.%%i >nul
)
arp -a | findstr "64-dd-e9"
```

Issues: Sequential, slow, may miss devices

Desired Solution

```batch
# Need: Parallel scanning with immediate MAC resolution
# Possibly using WMI or PowerShell jobs
```

Advanced Tool Integration Questions

1. WinPcap/Npcap Integration

Question: How to use packet capture libraries in batch/PowerShell for passive network discovery?

2. Windows Socket Raw Access

Question: Can we use raw sockets to send ARP requests directly?

3. UPnP/SSDP Discovery

Question: Can phone's FTP server advertise via UPnP for automatic discovery?

Performance Requirements

Current Performance

- Phone hotspot detection: 2-5 seconds
- Full network scan: 30-60 seconds
- Manual selection: 45+ seconds

Target Performance

- Any scenario: < 10 seconds

- Cached/known device: < 2 seconds

Security Considerations

1. **Firewall Rules**: Need automatic exception for FTP port 2121
2. **Credential Storage**: Secure storage for FTP password
3. **Network Scanning**: Avoid triggering IDS/security software

Alternative Approaches to Research

1. Static Configuration

- Configure router to always assign same IP to phone MAC
- Use Windows hosts file for name resolution
- Create network location awareness profiles

2. Phone-Side Solutions

- FTP server that announces presence (broadcast/multicast)
- Dynamic DNS update from phone
- WebDAV instead of FTP (better Windows integration)

3. Hybrid Approach

- Phone app that sends UDP broadcast with IP info
- PC listener service that catches broadcast
- Shared database/file with connection info

Specific Windows APIs/Tools to Research

1. **IP Helper API** (Iphlpapi.dll)
    - `GetIpNetTable` - ARP table access
    - `SendARP` - Force ARP resolution
    - `GetAdaptersInfo` - Network adapter details
2. **WMI Classes**
    - `Win32_PingStatus` - Async ping
    - `Win32_NetworkAdapterConfiguration`
    - `Win32_IP4RouteTable`
3. **PowerShell Cmdlets**
    - `Test-NetConnection` - Advanced connectivity testing
    - `Get-NetNeighbor` - ARP cache (Win 8+)

    - `Get-NetTCPConnection` - Active connections

4. **Native Tools Enhancement**
    - `wmic` queries for network info
    - `netsh` advanced commands
    - `sc query` for ICS service status

Testing Scenarios Needed

1. **Phone switches between networks** - How quickly detected?
2. **Multiple phones with FTP** - How to differentiate?
3. **VPN active** - Does it interfere with detection?
4. **IPv6 enabled** - Any complications?
5. **Airplane mode toggle** - Recovery time?

Requested Research Output

Please provide:

1. **Optimal WMI query** for real-time network device detection
2. **PowerShell script** for parallel network scanning with MAC resolution
3. **Registry locations** for Mobile Hotspot client information
4. **Native Windows method** for port checking without external tools
5. **Batch/PowerShell hybrid** for < 5 second device detection

Error Patterns to Solve

Pattern 1: "Phone not found" when it's connected

- Possible causes: ARP timeout, firewall, network isolation
- Need: Diagnostic steps to identify root cause

Pattern 2: "FTP connection refused" after detection

- Possible causes: Port mismatch, FTP server not running
- Need: Pre-connection validation method

Pattern 3: "Slow network scan" on large networks

- Possible causes: Sequential processing, timeout values
- Need: Parallel scanning algorithm

Success Criteria

The solution should:

1. Detect phone in < 10 seconds in any network configuration
2. Work without administrative privileges
3. Not trigger security software
4. Handle network changes gracefully
5. Provide clear diagnostic information on failure
6. Support multiple simultaneous FTP phones (future requirement)

Additional Context for AI Research

- Windows 10/11 environment
- No installation of system services preferred
- Batch file primary, PowerShell secondary
- Must work with Windows Defender enabled
- Cannot modify phone's FTP server configuration
- Solution should be portable (no registry changes)

Please research and provide solutions for the technical challenges listed above, focusing on native Windows capabilities and minimal dependencies.