1 Device Discovery on Lan → Browsers cannot scan lan directly.

↳ mDNS (multicast DNS) : Allow automatic detection of devices. [To get list]

↳ NetBIOS or ARP Scanning : Find LAN devices via backend code

2 Hosting a local server for communication

Browsers cannot directly scan networks.
You need a local Node.js server.

Node.js Express Server (LAN Messaging)
Web Client to Discover LAN Users

3 Real-Time Messaging with WebSockets:

Once users are discovered, use websockets for real-time communication

WebSocket Server (Node.js)
WebSocket Client (Frontend JavaScript)

4 Peer-to-Peer Communication (No Server)

If you don't want a central server, use WebRTC for direct LAN connections.

Simple WebRTC (using simple-peer)

```
const peer = new SimplePeer({ initiator: location.hash === "#host" });
peer.on("signal", data => {
    console.log("Signal Data:", JSON.stringify(data));
});
```

RANGES

1. Local

Your s
which

Go

2. He

↳ A

↳ on

↳ Ne

↳ Br

↳ W

↳ W

Mar 18, 2025, 22:42

Ranges Of Search

1. Local Network Range (IPv4)

Your search is limited to devices within the same subnet of the LAN, which is typically:-

* Ethernet : upto 100 meters (physical cable limit)
* Wifi : 30-100 meters (depends on the router's signal strength).
* Campus Network : If all routers are part of the same LAN, it can cover the entire campus.

To find devices in the same subnet, you can scan the local IP range (eg. 192.168.1.0/24 or 10.0.0.0/24).

2. Methods & their Range

Method	Discovery Range	How it works
↳ ARP Scan	Entire Subnet (LAN)	lists all active devices on the same network.
↳ mDNS (multicast DNS)	1 Router Hop	Detects services on the local network (like Apple AirDrop).
↳ NetBIOS/NBTSCAN	Entire Subnet	Finds Windows devices on the LAN.
↳ Broadcast Ping	Entire Subnet	Sends a ping to all devices (eg. ping 192.168.1.255)
↳ WebRTC (P2P)	User-to-User only	Direct peer-to-peer within the LAN requires known IPs of two
↳ WebSockets (Server-Based)	Entire LAN	Needs a local WebSocket server to relay messages.

3) Expanding the Search

If the lan has multiple subnets, you might need:

- Subnet Scanning (192.168.0.0/16 to scan wider range)
- Routers with Multicast Support (for mDNS across subnets).
- Custom Local Server that bridges different subnets.

Mar 18, 2025, 22:42

DISCOVERY

# Scenario	Solution
↳ Same Room Wifi/LAN	– Multicast UDP (faster & local)
↳ Different Room (same college LAN, Different Subnet)	– Try Multicast → If blocked, use Websocket Relay
↳ Different Hostel (different LANs)	– Websocket Relay (If routing allows)

This method ensures no Internet is needed and works across all cases.

METHODS:-

1. Multicast UDP for Local Discovery (same subnet)
   - ↳ Each device sends a multicast "Hello" message on the LAN.
   - ↳ Other devices listen and respond allowing discovery.
2. Websocket Relay for Cross-Router/Hostel Discovery
   - ↳ If multicast doesn't work across subnets, use a local websocket server on a LAN computer.
   - ↳ Devices register with the server and get a list of connected peers.

FUNCTIONALITY

Services	Methods
1. Voice/Video chat	– WebRTC P2P
2. Text & File sharing	– (TCP/UDP Sockets)
3. Group Chat	– (LAN Websocket Server)

Methods:-

1. WebRTC for
   - ↳ WebRTC
   - ↳ Once
   - ↳ If they

Steps for
1. A
2. Once
3. If

2. TCP/UDP
   - ↳ For text
   - ↳ Works

3. Websocket
   - ↳ For
   - ↳ Works

For Running
   - ↳ For 24/
   - ↳ If you

Mar 18, 2025, 22:42

Methods:-

1. WebRTC for Direct Peer-to-Peer Communication
   - ↳ WebRTC works without internet if you use a LAN-based signaling server.
   - ↳ Once connected, peers can talk, text, or share files over LAN.
   - ↳ If they are in the same subnet, it works P2P without a relay

   Steps for WebRTC over LAN:-
   1. A local signaling server helps peers find each other.
   2. Once connected, WebRTC sends data/audio/video directly between peers.
   3. If peers are on different subnets, they may need a relay.

2. TCP/UDP Socket Communication (Text & Files Over LAN)
   - ↳ For text messaging or file sharing without internet use TCP or UDP sockets
   - ↳ Works even if devices are in different subnets.

3. WebSocket Server for LAN-wide Communication
   - ↳ For multi-user chat, use a WebSocket server inside the LAN.
   - ↳ Works across different LAN segments if routing allows.

For Running local Server (Hosting) over Lan

- ↳ For 24/7 availability, use a Raspberry Pi as a LAN server.
- ↳ If you need more power, use a mini PC or an old laptop

Mar 18, 2025, 22:42

REQUIREMENTS FOR LOCAL SERVER

i) For 5000 users → Powerful Hardware [16-core CPU, 32 GB+ RAM, 1-10 Gbps Network]

ii) Strong Lan Setup → Enterprise router, VLANS, static IP.

iii) Load Balancing (NGINX ⚙, HAProxy)

PREREQUISITES

1) Technical Knowledge

i) Networking Fundamentals:-

* IP Addressing & Subnets :- Understanding IPV4 addressing (eg. 192.168.x.x) and subnetting.

* DHCP vs Static IP :- Configure static IP addresses for the server on the local network.

* LAN Setup :- Setting up routers, switches and VLANS (for traffic segmentation)

* Port forwarding :- Understand how to open and route specific ports if needed.

* Network Troubleshooting :- Use tools like ping, traceroute, and netstat for diagnostics.

ii) Backend Development

* RESTAPI Development
* Websockets
* Process Management : use tools like pm2 for (Node.js) to manage and restart services.
* Asynchronous Programming : For handling multiple users concurrently (eg. using async/await)

Mar 18, 2025 22:43

(iii) Database Management

* Connection pooling :- Efficiently manage database connections
* Indexing & Query Optimization :- Reduce query time for large datasets
* Data Backup & Restoration :- Regularly backup critical data.

(iv) System Administration

* File Management :- Navigate and manipulate files using commands (ls, cp, mv, etc)
* Process Management :- Monitor and control server processes using htop, systemctl.
* Security :- Implement firewalls (ufw), SSH access, and user permissions
* Logging :- Set up log monitoring using tools like journalctl, /var/log

(v) Performance Optimization

* Caching :- Implement caching strategies (eg. Redis or in-memory caching)
* Load Balancing :- Use NGINX or HAProxy to distribute traffic across multiple instances
* Profiling & Monitoring :- Use tools like prometheus and Grafana for the metrics.
* Benchmarking :- Simulate traffic using Apache Benchmark (ab) or locust.

2. HARDWARE REQUIREMENTS

NOT TO WORRY FOR NOW

3. Tools & FRAMEWORKS :-

* Docker :- For containerizing and scaling your app.
* NGINX :- As a reverse proxy for load balancing.
* Git :- Version control for your codebase.
* Monitoring :- Prometheus, Grafana, or Netdata for server health tracking

4. TESTING & DEPLOYMENT :-

* Benchmarking Tools :- Use ab or wrk to test server load.
* CI/CD :- Automate builds and deployments (eg github Actions)
* Backup Strategy: Implement regular backups for database and critical data.

Mar 18, 2025, 22:43