GenosDB/GenosRTC

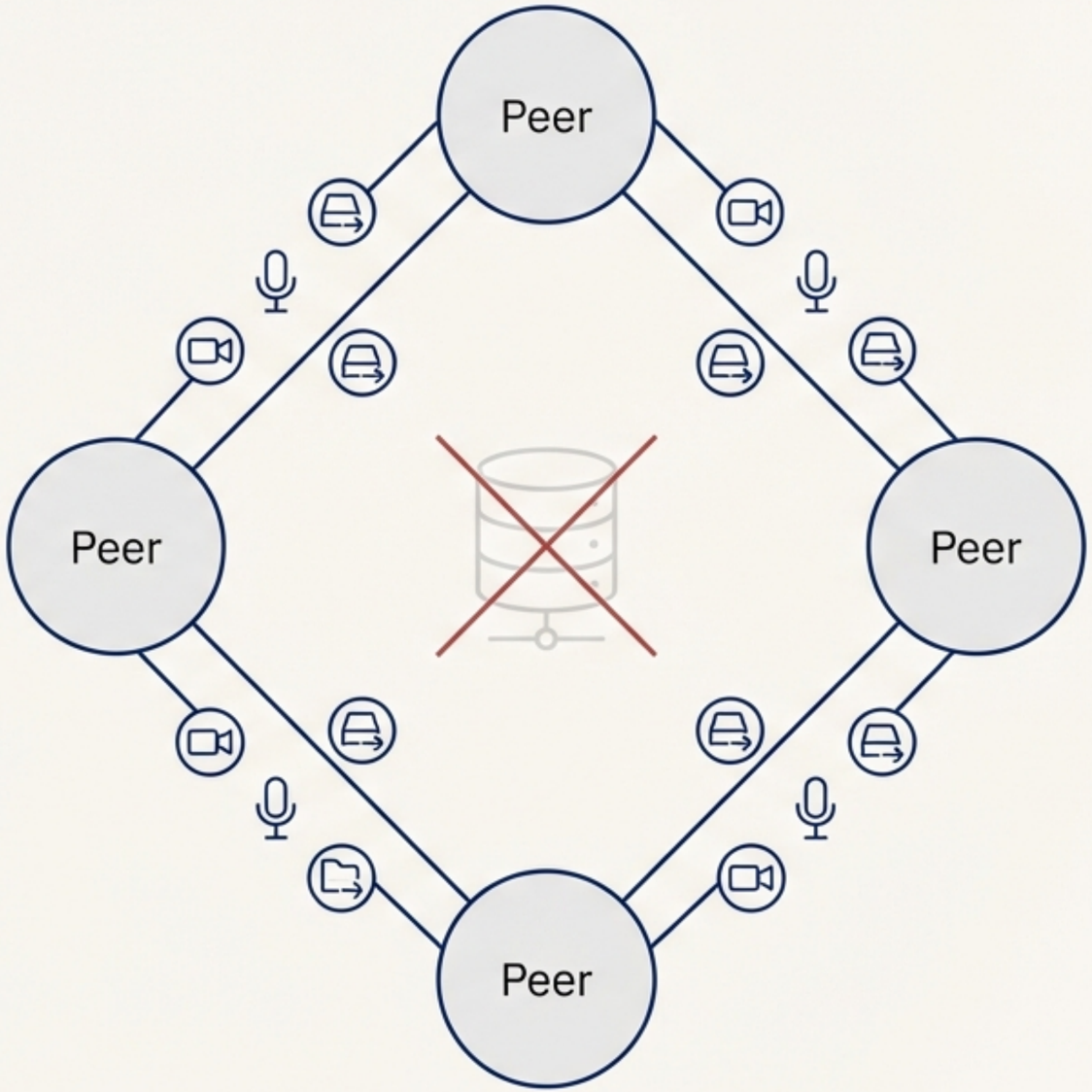
GenosRTC: Scaling WebRTC from Dozens to Millions

An Architectural Deep Dive into the Self-Organizing Cellular Mesh Protocol

WebRTC empowers true peer-to-peer communication

GenosRTC is a high-level P2P module integrated within the GenosDB ecosystem. Its core architectural goal is to deliver a ‘serverless’ communication model where peers connect directly to one another.

- **Low Latency**: Data and media flow directly between peers.
- **Enhanced Privacy**: Eliminates centralized data-handling servers.
- **Resilience**: No central point of failure or data bottleneck.



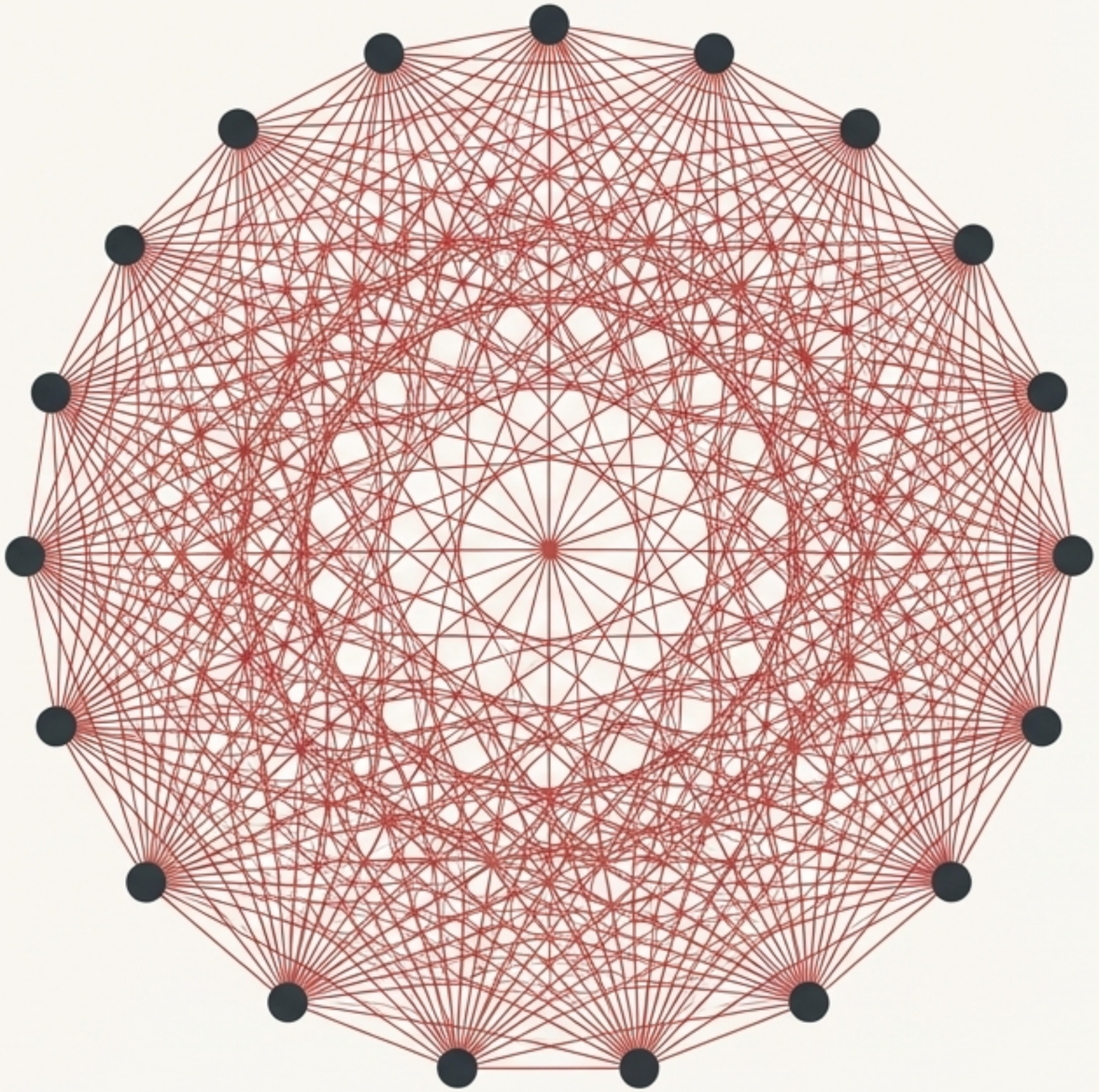
The standard mesh network breaks under pressure

In a full mesh, every peer connects to every other peer. This creates an $O(N^2)$ connection problem, leading to “network flooding” as each peer retransmits messages to all neighbors. This exponential overhead limits most WebRTC applications to around 100 concurrent peers.

The Connection Formula

```
N Peers = N * (N-1) / 2 Connections
```

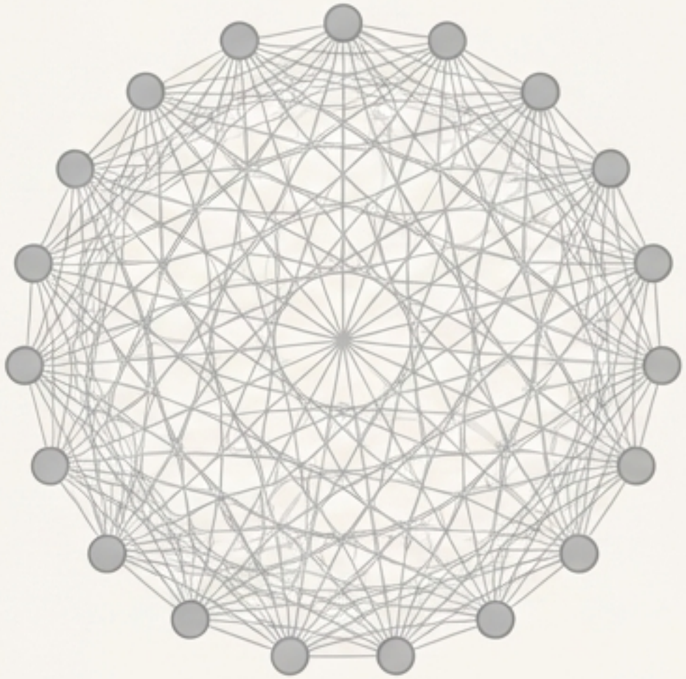
100 Peers = 4,950 Connections



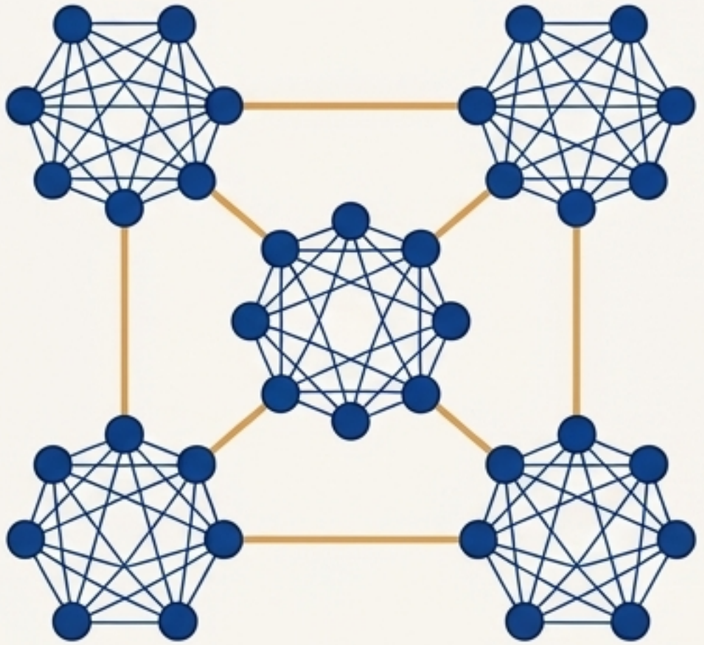
From a single giant mesh to a network of nimble cells

Instead of one massive, unmanageable mesh, GenosRTC organizes peers into smaller, interconnected ‘cells’. Specialized ‘bridge’ nodes intelligently route traffic between these cells, creating an adaptive ‘network of networks’ that prevents network-wide floods.

Standard Mesh



Cellular Mesh



The building blocks: Cells, Peers, and Bridges

Cells

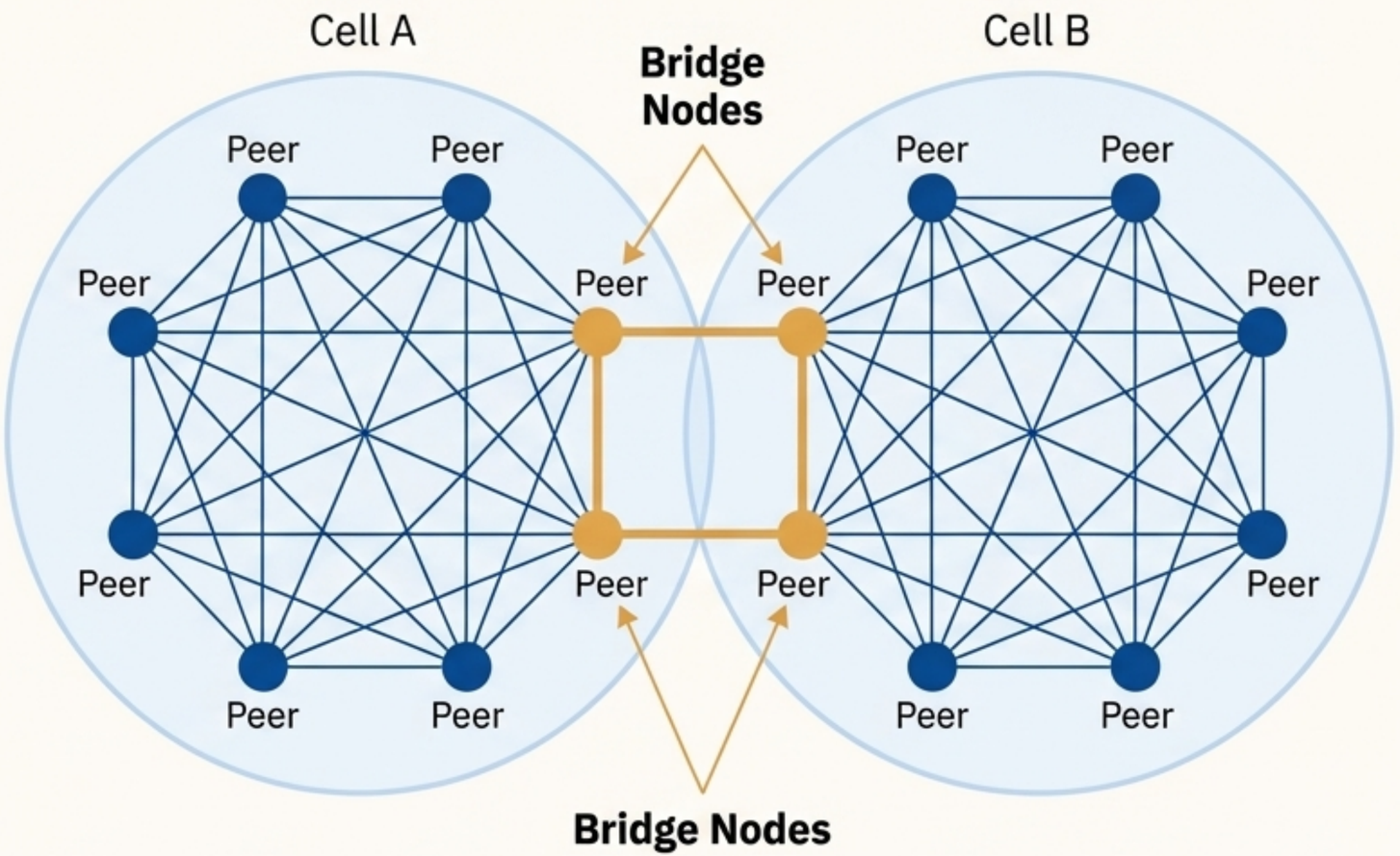
Small, fully-meshed groups of peers (typically ~20-50). High performance and low latency are maintained locally within the cell.

Peers

Assigned to cells via consistent hashing on a hash ring. This ensures network stability as peers join and leave (‘churn’).

Bridges

Specially elected peers on the edge of adjacent cells. Their sole function is to connect cells and forward inter-cell traffic. Multiple bridges provide redundancy.



Smart messaging avoids network-wide floods

Intra-cell (Local First)

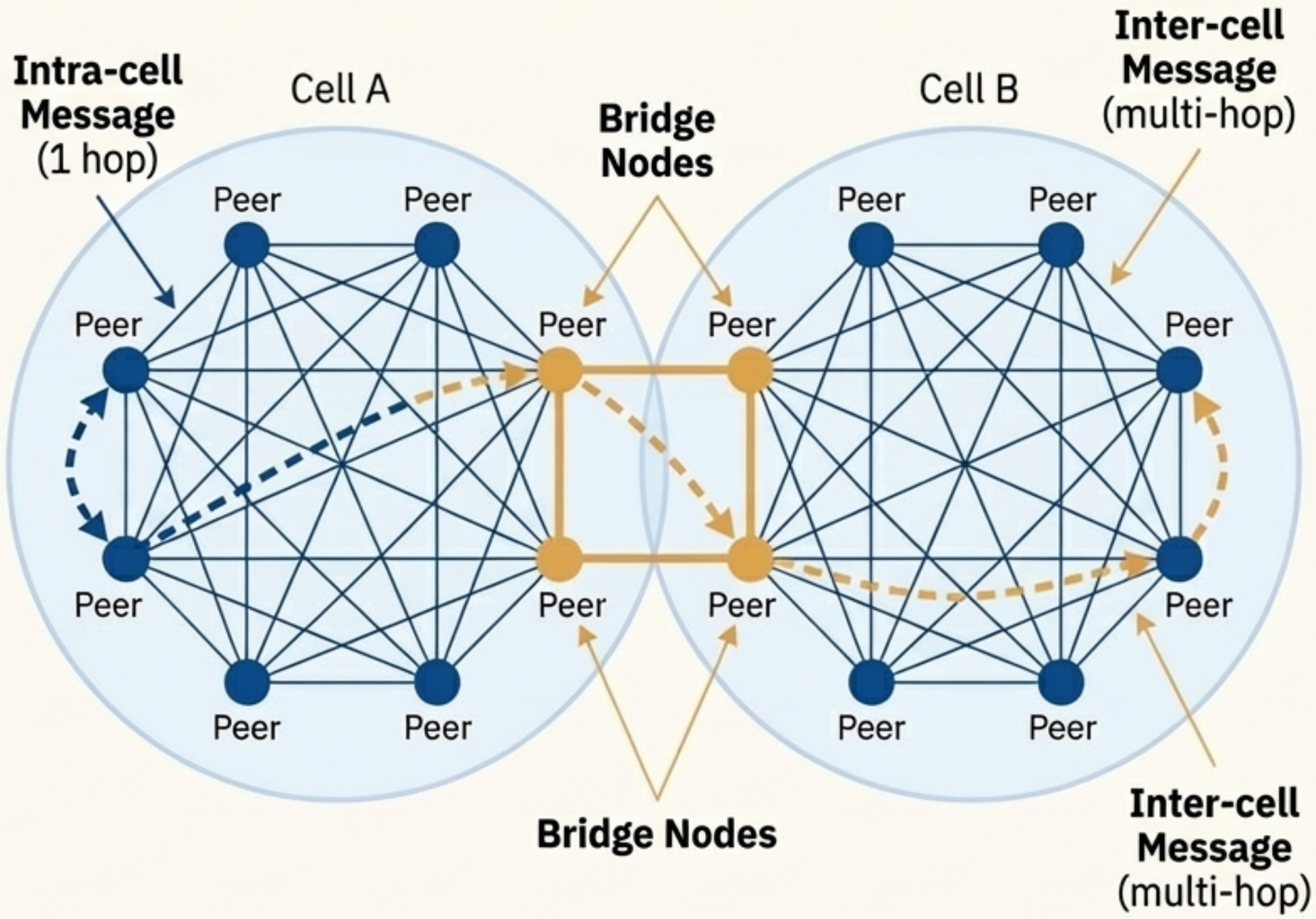
Messages destined for a peer within the same cell are delivered directly in a single hop, just like a small mesh.

Inter-cell (Global When Needed)

Messages for other cells are passed to a bridge node, which forwards them to the next cell.

Infinite Loop Prevention

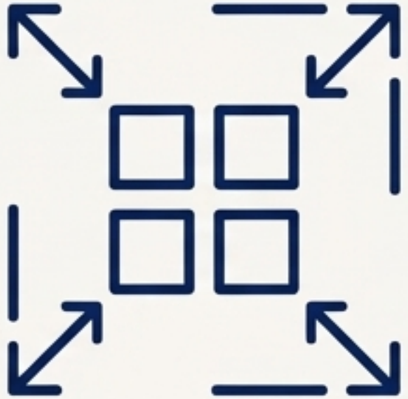
A built-in ‘Dynamic TTL’ (Time-To-Live) acts as a hop limit, ensuring messages expire. It is calculated automatically based on network topology (e.g., `2 × numberOfCells + buffer`).



The network is a living, adaptive system

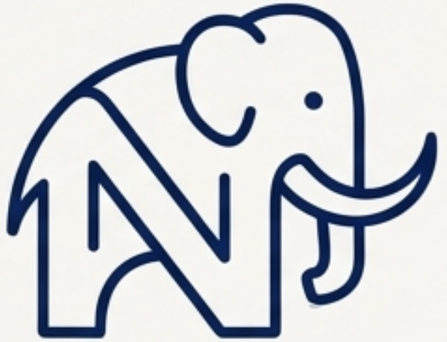
Health Scoring & Intelligent Election

Bridges aren’t random. They are elected from the most stable peers based on a `healthScore` calculated from uptime, average latency (RTT), and recent activity.



Dynamic Cell Size

When set to `'auto'`, the optimal `cellSize` is calculated automatically based on the total number of peers, adapting to network growth (capped at `maxCellSize`).

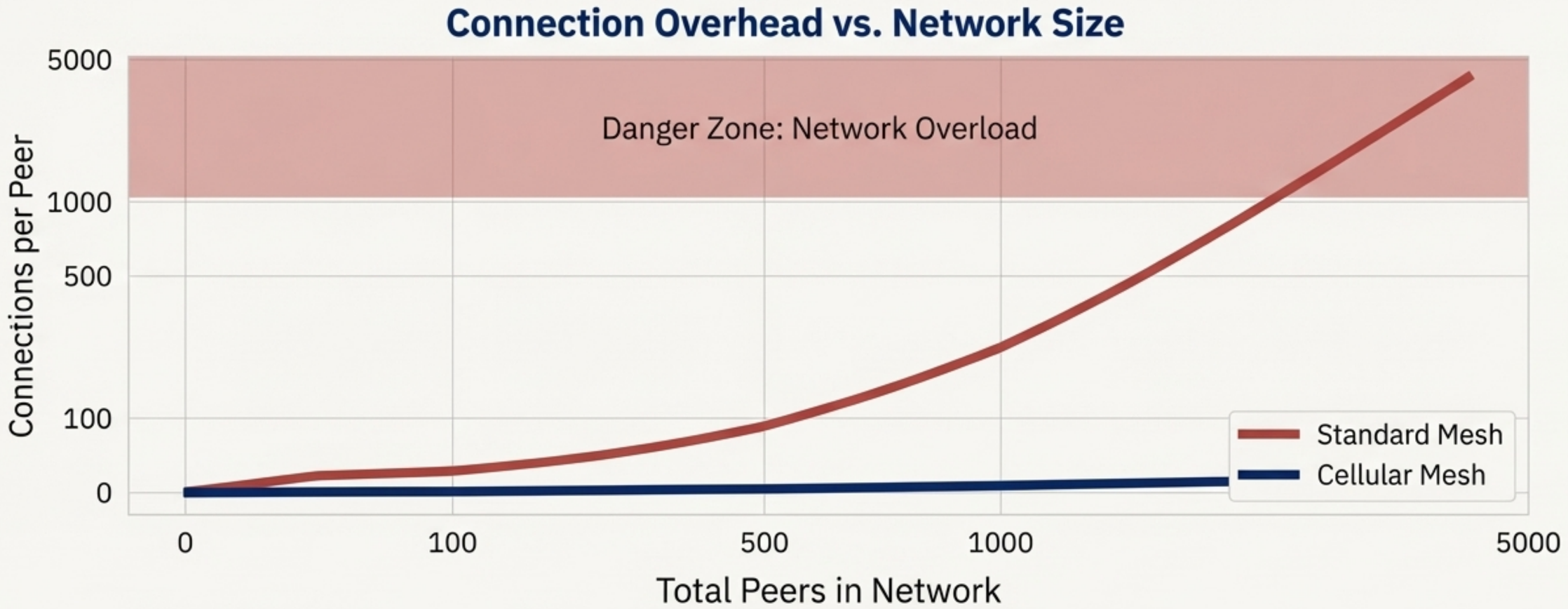


Decentralized Signaling

Peer discovery is handled by the resilient and community-driven Nostr network, avoiding central points of failure and infrastructure overhead for developers.

From exponential collapse to linear scale

The Cellular Mesh transforms the connection overhead from an exponential $O(N^2)$ problem to a manageable, linear one.



Drastically reduced load, massively increased capacity

Metric	Standard Full Mesh	Cellular Mesh (10k peers)
Connections/peer	~10,000	~100
Message hops (avg)	1	~10
Max practical peers	~100	Massive scale

A 100x reduction in connection overhead for a 10,000 peer network.

Activating massive scalability is a single line of code

GenosRTC’s high-level API abstracts away the complexity of the underlying network topology. Developers can switch between modes with a simple configuration flag.

For < 100 Peers (Standard Mesh)

```
gdb('dbName', { rtc: true });
```

For 100+ Peers (Cellular Mesh)

```
gdb('dbName', { rtc: { cells: true } });
```

Choose the right topology for your use case

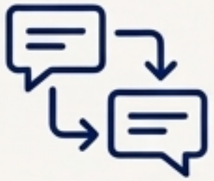
The best configuration depends on your application's expected concurrency.
Use this guide for recommendations.

Use Case	Expected Peers	GDB Configuration
Small Team Chat / Collab	< 100	`rtc: true`
Large Webinar / Event	100 - 1,000+	`rtc: { cells: true }`
Massive Multiplayer Game	1,000+	`rtc: { cells: { bridgesPerEdge: 3 } }`
IoT / Sensor Network	5,000+	`rtc: { cells: { targetCells: 200 } }`

The entire GenosRTC API runs on the Cellular Mesh

Enabling the Cellular Mesh enhances the scalability of the entire P2P feature set. All high-level communication abstractions, from data channels to media streams, operate seamlessly on the new topology.

Powered by Cellular Mesh



Data Channels (`db.room.channel`)

Audio Streams (`db.room.addStream`)

Video Streams

File Transfers

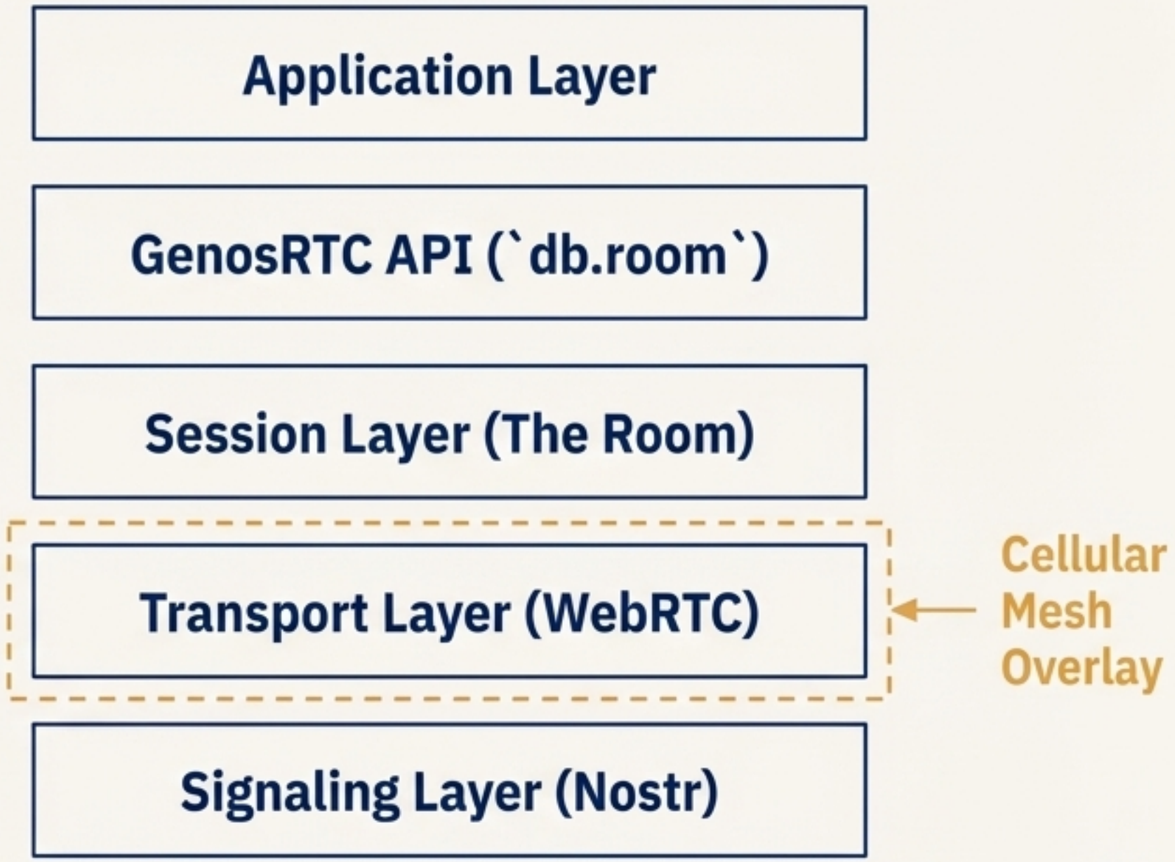
A layered architecture for robust, decentralized applications

Decentralization First
Peer-to-peer data exchange with no central bottlenecks.

Simplicity through Abstraction
A clean, event-driven API (`db.room`).

Room-Based Scoping
P2P interactions are scoped within a logical ‘room’.

Secure by Design
Mandatory transport encryption (DTLS/SRTP) and optional E2EE.



Redefining the limits of real-time P2P

Solves the critical $O(N^2)$ scalability problem inherent in standard WebRTC mesh networks.

Enables massive-scale applications with thousands of concurrent peers through a self-organizing, self-healing cellular network.

Maintains a simple, high-level API, hiding complexity and making massive scalability accessible via a single line of code.

Built on decentralized principles, using Nostr for resilient signaling and ensuring privacy with end-to-end encryption.

GenosDB/GenosRTC

Get Started with Scalable WebRTC

Read the Docs*

Link to API Reference & Architecture Docs

Explore the Code*

Link to GitHub Repository

Start Building*

```
gdb('my-large-app', { rtc: { cells: true } });
```