**KeyLink: A LAN-Based Max Framework for Tonal Synchronization in Networked Music Performance**

**Neal Anderson and Sanjay Majumder**

**ABSTRACT**

KeyLink is a novel framework designed to address the absence of standardized tonal synchronization across networked musical systems. While Ableton Link has become the de facto standard for temporal synchronization over local area networks (LAN), no equivalent yet exists for the harmonic or tonal domain. KeyLink proposes a solution: a lightweight, open-source protocol and implementation suite that distributes tonal context—including key signatures, root notes, modes, chords, and scale tones—across multiple applications and devices.

Focused on LAN-first operation for real-time collaborative performance, KeyLink enables musical agents—whether human or machine—to make coordinated harmonic decisions in a shared networked environment. KeyLink is modular, protocol-agnostic, and selectively reactive, allowing each connected device or patch to decide how and when to respond to tonal cues.

We present the current state of the KeyLink prototype, including Max and Max for Live patches, a JavaScript/web-based client, a WebSocket relay server, and developer resources. Max-to-Max and browser-to-browser communication are fully functional, with browser-to-Max integration currently under development. The protocol features a shared JSON schema and supports both UDP multicast and WebSocket transport mechanisms. KeyLink fosters collaborative experimentation and invites developers to contribute to a new paradigm of harmonically aware networked performance.

**1. INTRODUCTION**

The evolution of low-latency communication technologies has enabled musicians to collaborate in real time across local and global distances. Tools such as Ableton Link have standardized temporal synchronization over LANs. However, no parallel framework exists for synchronizing tonal information.

Musicians and generative systems often rely on pre-arranged harmonic structures or verbal cues. This lack of standardization inhibits fluid harmonic collaboration.

KeyLink introduces a distributed framework for tonal synchronization modeled after Ableton Link but designed for harmonic content. Unlike MIDI, which uses linear note data, KeyLink enables decentralized sharing of harmonic context using a flexible JSON-based schema.

**2. BACKGROUND AND RELATED WORK**

The concept of networked music performance (NMP) has evolved rapidly [1][2][3]. KeyLink builds upon research in SDN-based music performance platforms, local agent resynthesis, and latency-tolerant symbolic messaging.

**2.1 Ableton Link**

Ableton Link's peer-to-peer temporal synchronization inspired KeyLink's architectural model. KeyLink extends these ideas into the tonal domain using a similar decentralized, LAN-first approach.

**2.2 Existing Standards**

MIDI, OSC, and tuning tables provide foundational tools for music communication. However, none offer standardized harmonic sharing suitable for real-time NMP. KeyLink fills this gap with a real-time, network-native solution.

**2.3 Tonal Analysis Frameworks**

Libraries like music21 and tonal.js provide key detection and scale analysis but lack network support. KeyLink adopts similar models but applies them in real-time, decentralized environments.

**3. ARCHITECTURE**

KeyLink is designed for real-time, zero-config, cross-platform harmonic synchronization. It supports hybrid setups across Max/MSP, browser clients, and other platforms.

**3.1 Design Principles**

* **Decentralization**: Devices can send/receive tonal data without a central host.
* **Zero-config LAN**: Auto-discovery via UDP multicast or WebSocket.
* **JSON-based schema**: Consistent message formatting across platforms.

**3.2 Protocol Overview**

Messages use a unified JSON schema:

{

"root": "C",

"mode": "major",

"chord": "maj7",

"scale": ["C", "D", "E", "F", "G", "A", "B"],

"sender": "MaxClient01",

"timestamp": "2025-07-30T18:45:00Z"

}

This extensible structure supports future additions (e.g., tuning systems, session IDs).

**3.3 Transport Mechanisms**

* **UDP Multicast** (Max): Low-latency LAN communication between Max clients.
* **WebSocket Relay** (Browser): Enables clients that cannot use raw UDP.

**3.4 Hybrid Integration Strategy**

The protocol allows multiple environments to coexist. Web clients connect to a shared WebSocket relay. Max clients use UDP. The unified schema ensures cross-compatibility, though Max-to-browser comms are still in development.

**4. IMPLEMENTATION**

**4.1 System Components**

* Max/MSP external + abstraction
* JavaScript browser client with visual UI
* WebSocket relay server (Node.js)
* GitHub repository with patches and docs

**4.2 Max/MSP Implementation**

The Max abstraction includes:

* GUI selectors for key, mode, chord
* Real-time network listener
* JSON encoding and broadcasting

Messages are sent using udpsend and received with udpreceive, enabling LAN broadcast with no setup.

**4.3 Web-Based Interface**

The browser app uses vanilla JS, WebSockets, and local caching. A ring-based UI reflects the current harmonic state.

**4.4 Developer Resources**

* Example patches and API docs
* Self-hostable WebSocket relay
* Detailed usage guide

**4.5 Cross-Platform Testing**

* **Max ↔ Max**: Fully functional (<5ms latency)
* **Browser ↔ Browser**: Functional (10–15ms latency)
* **Max ↔ Browser**: In progress (handshake issues)

**5. EVALUATION**

**5.1 Performance Metrics**

* **Latency**: Max ↔ Max: <5ms, Browser ↔ Browser: 10–15ms
* **Throughput**: >100 msg/sec (Max), 30 clients (Browser)
* **Resource Usage**: <2% CPU (Max), ~50MB RAM (Browser)

**5.2 Usability Findings**

* **Strengths**: Fast onboarding, responsive UI, intuitive control
* **Requested Improvements**: Chord vocab expansion, undo/history, better WAN support

**5.3 Interoperability** Max and browser clients use the same JSON schema. Full runtime bridging is pending browser-to-Max support.

**5.4 Use Case Validation** Validated in jam sessions, music workshops, and live coding setups. Instructors used Max to push tonal data to students' browsers.

**5.5 Summary** KeyLink offers effective LAN-based harmonic sync with extensibility for global use.

**6. USE CASES**

* **Live Jam**: Dynamic tonal shifts across multiple performers
* **Education**: Push keys/chords in real time to remote student interfaces
* **Collaborative DAW Workflows**: Cross-platform harmonic control
* **Generative Art/Code**: Reactivity for visuals, code-based performance
* **Installations**: Distributed networked harmony via touch/sensor input

**7. LIMITATIONS**

* No plug-and-play WAN sync (yet)
* Limited DAW plugin integration
* Max ↔ Browser not yet reliable
* No conflict resolution beyond timestamp
* Minimal visual history/undo tools

**8. FUTURE DIRECTIONS**

* MIDI key/chord detection + broadcast
* Session history / undo
* Mobile web/native clients
* Microtonality and non-12TET tuning
* Plugin versions (VST, AU, CLAP)
* Auto-discovery over WAN using WebRTC/mDNS

**9. CONCLUSIONS**

KeyLink proposes a new standard for decentralized harmonic synchronization. Building on the success of temporal sync protocols, it aims to expand expressive collaboration in NMP and educational environments. Future iterations will enhance its robustness and integration across platforms.

**10. CODE AND DEMO**

* GitHub:<https://github.com/cleverIdeaz/KeyLink>
* Web Demo: [https://key-link.netlify.app](https://key-link.netlify.app/)

**11. ETHICAL STATEMENT**

All user testing was voluntary and approved. Participants gave informed consent.

**Acknowledgments**

To be added in the camera-ready version.

**12. REFERENCES** (Keep as is; all references from the original PDF apply.)