GlobeMusic: The Internet Scale of eMusic-Making

Max Mühlhäuser
FG Telekooperation
TU Darmstadt
Alexanderstr. 6
64283 Darmstadt, Germany
+49 6151 16 3709
max@informatik.tu-darmstadt.de

Michael Welzl
Telecooperation Department
University of Linz
Altenbergerstr. 69
4040 Linz, Austria
+43 732 2468 9264
michael@tk.uni-linz.ac.at

Jan Borchers
CS Department
Stanford Universaty
Gates Building 2a-254
Stanford, CA 94305-9020, USA
+1 650 725 7646
borchers@stanford.edu

Rainer Gutkas
Telecooperation Department
University of Linz
Altenbergerstr. 69
4040 Linz, Austria
+43 676 3268369
rainer.gutkas@jk.uni-linz.ac.at

ABSTRACT
The paper is centered around new musical genres and new roles of musical content providers and consumers in the upcoming fourth mass medium: the Internet. We introduce GlobeMusic, a system and concept for eMusic-Making in this context. Major ingredients of GlobeMusic were developed and exposed to a large public in the last four years. GlobeMusic and its components mainly address the following issues: computer-human interaction, individual eMusic-making, appropriate computer network protocols, musical data formats, and cooperative eMusic-making.

Keywords
Internet music, HCI, computer-based instruments, cooperative eMusic-making.

1 INTRODUCTION

The present paper discusses and proposes novel approaches to music-making in the light of the breathtaking advancements of the Internet. We will describe several systems and concepts developed over the last four years – in part inedited – and conclude by describing a system called GlobeMusic which integrates and augments the different aspects and concepts introduced in the course of the paper. Please note that while the full-fledged GlobeMusic system is still to be finalized, virtually all its parts have already been evaluated, some even used by a large number of users;

The field of "Internet eMusic-Making" addressed is highly dynamic and each of the ingredients of GlobeMusic may contribute to the advancement of this new and fascinating field.

The Internet *is* a computing and telecommunication infrastructure for the masses and it is *about to become* the 4th mass medium after press, radio, and TV. The present paper addresses this latter issue for the domain of music, emphasizing the following perspectives: computer-human interaction, individual music-making, computer network protocols, music data formats, and cooperative music-making.

In our work, we found it crucial to accept the following statement*: like any new mass medium, the Internet is going to foster new content types (genres).* Readers who doubt here may be referred to the fact that for the case of video, e.g., Santa Monica has become a melting pot of ingenious people and companies who are about to investigate new content types and, of course, new ways of creating them [6] – many of them left Hollywood because they felt that this famous site was too much bound to the existing genres. Two basic assumptions of the protagonists of Internet-specific genres are worth noting: i) Internet genres will be highly interactive; ii) the boundaries between content producers and content consumers will become much more blurred. While we will concentrate on music instead of video below, we will adopt these very same assumptions for research about *new Internet-specific musical genres.*

Of course, new genres emerge from a process that is influenced by a highly dynamic and non-predictable gravity field with three poles: the (changing) taste, budget, and interest of consumers; the (improving) limits of technical feasibility; and the (least predictable) genius and fantasy of content creators. Therefore, the present paper does not claim to "know" the nature of future musical genres, but rather reports on systems and concepts developed, experiments, and large-scale user experiences as elements of the melting pot from which new genres will eventually emerge. Fig. 1.1 depicts a scenario of several musicians connected

to the Internet (potentially from different corners of the globe) and to a distributed computer-based infrastructure (depicted with a "hard disk" symbol) for cooperative eMusic-making.

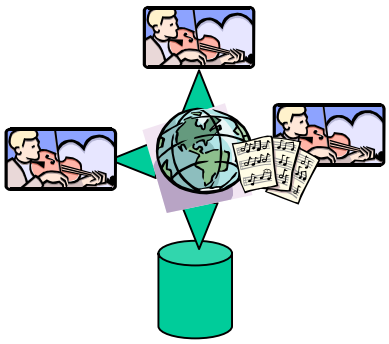

Fig. 1.1 Global eMusic-Making Scenario

Overview

On one hand, we distinguished creators and consumers of contents above, on the other hand we argued that their roles might become blurred - how does this match? The following aspects must be considered for better understanding:

- In the case of music, the three-step process of composition, interpretation, and listening has long been generally accepted. However, if a Jazz band unites for a jam session without additional audience, they are supposed to have fun being (largely) composing, interpreting, and listening all in one. This shows that even without the Internet, modern music-making tends to blend formerly distinct roles.
- The Internet is not only the first mass medium to be interactive, it is also the first one which supports goes beyond broadcast by embracing person-to-person and group-to-group communication. This fact contributes essentially to the move from merely consumptive to partly creative "consumers".
- Envisioning the Internet as an interactive medium for the masses, we face dramatic implications: couch-potatoes and other mere consumers are supposed to become interactive, even creative! This is only possible if the interaction/creation process is easy to carry out yet highly attractive (fun, motivating, thrilling, ...) – readers who doubt this argument in general should note that i) interactive computer games do represent a booming *mass* market; ii) the authors gained very positive experiences exposing average users to interactive, even creative musical Internet-Instruments. This implies that the "producer" side in the media business moves from *authoring read-only media* via *creating interactive media* towards *software development for 'create-your-own' toys/tools*. Referring to the classical world of music, one could rephrase this process as: "the users embrace composition, interpretation, and listening all in one, while the former composers shift to the task of inventing ever new (software/hardware) instruments".

In the remainder of this paper, we will describe the following four systems and approaches that we developed over the past four years – each addressing a different issue of GlobeMusic as depicted in fig. 1.2. The steps towards the final system are coined with different "proper names" since these steps were exposed to the public as ready-to-use system for experimentation. The steps (and systems), in chronological order, were called WorldBeat → NetMusic → NetScore-RT → GlobeMusic.

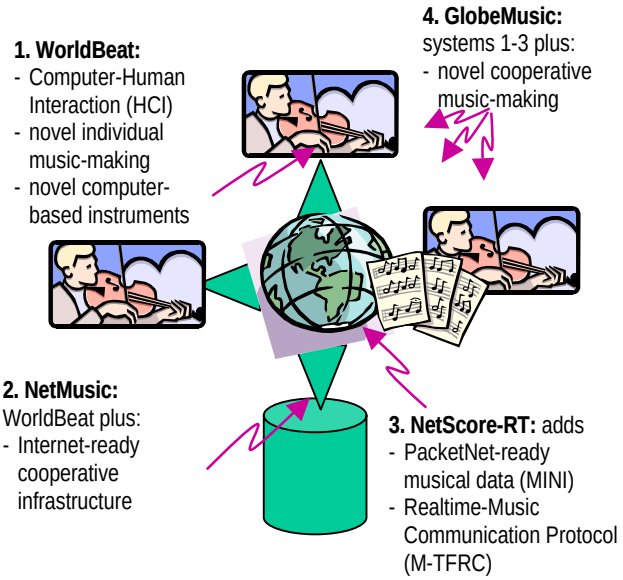

Fig. 1.2 GlobeMusic components related to fig. 1.1

2 WORLDBEAT: BREAKING DOWN THE ACCESS BARRIER

Networked music creation systems aim at creating an online community of users that engages in social interaction by cooperatively playing, improvising, and performing music. Therefore, their success depends critically on a broad user base. The first step to building a GlobeMusic system then is to ensure that it appeals to a broad body of potential users across the global network.

Some requirements for music systems targeting an Internet-wide user base

In classical desktop computing scenarios, such as running office applications, users are more inclined to accept a steep learning curve in exchange for a long-term payoff in work efficiency. In systems that aim primarily at supporting social interaction, such as GlobeMusic, however, those steep learning curves have to be avoided to give first-time, short-time users a chance of success in their interactive experience. Ideally, such a system should scare off neither computer novices nor music novices to maximize the potential user base.

To take down the barriers that prevent potentially interested users from interacting with GlobeMusic, its user interface and functionality then have to follow a set of rules that can be derived from the design of interactive music exhibits and similar public-access systems [2]:

Simple Impression: The user interface has to make clear that it is going to be easy to use. In particular, if it is supposed to be a system about music, it should not appear to be a "complicated application" that requires a computer expert to operate it.

Innovative, Physical Appearance: Despite simplicity, the system has to suggest (and deliver!) a novel kind of interactive experience that is more physical and "realistic" than using the standard mouse and keyboard.

Balancing Creativity and Support: On the one hand, it is important for the user to receive help from the system when creating music using it. However, this help and guidance has to be supplied in a subtle way that ensures that the user does not feel decoupled from the creative process.

Computer-Supported Improvisation

The above requirements can be fulfilled by a combination of innovative user interface technology and metaphors, with software functionality that carefully models musical concepts to assist the player in the creative process.

We have developed a successful solution for the first part, supporting the user in creating input to a GlobeMusic system, by building a software component that carefully adjusts the user input in real time in order to ensure a satisfying result independent of the user's musical "abilities". When we exposed this part of GlobeMusic to the audience as a self-contained system, we coined it as *WorldBeat.* In general, there are three major dimensions of musical performance in which a system can try to support the user, but we will explain why we chose to control only one of them:

The rhythmical dimension (adjusting the timing of user input to match a given beat; generally called "quantizing"). While it is computationally relatively easy to "straighten" musical input to match a fixed pattern, this obviously leads to a machine-like rendition of the input. This can be improved upon by building components that take certain rhythmic irregularities into account, to create a more "human-like" quantization. Nevertheless, rhythmic support always leads to two problems: If the input is to be processed in real time, musical events can always only be delayed to a later point in time, never shifted "forward" to an earlier point. Second, and far more substantially, however, any kind of rhythmical adjustment will invariably lead to the user feeling decoupled from the creation process: If a user "plays" a note, and that note sounds not immediately but slightly later to match the beat pattern, the user will feel she is no longer playing an (augmented) instrument, but instead only controlling the process that actually creates the notes. Therefore, rhythm is not our preferred choice dimension to interfere with the user input.

The melodic dimension (adjusting the melodic patterns of input). This would mean, for example, creating whole melodies in response to a single user input event. While this technique has found wide adoption in consumer music devices (the infamous "arpeggiator" on the average home keyboard comes to mind), we believe that it again makes the user feel like he is losing a vital aspect of control over what is happening musically. The satisfaction of triggering entire musical sequences with the press of a key quickly fades, when users discover that this makes it increasingly difficult to tell the user's creative input apart from what the system is generating on its own. Therefore, we leave the melodic dimension to the user as well: He can play high or low notes, runs or chords, and the system only plays one note per user-level input event, much like a traditional instrument.

The harmonic dimension (adjusting the harmonic structure of the input). This means taking the user input, and remapping it slightly (changing pitches) to fit the current harmonic context. This is a very suitable dimension for augmentation, since the user can still control the rhythm and melodic profile of his performance. The system only takes the input and changes it just so that it fits into the harmonic structure of the accompaniment. This is what our ImprovisationHelper does. Another advantage of this approach is that it works perfectly even on real-time input that requires real-time output of the augmented data stream.

To understand how the component "ImprovisationHelper" works in detail, we will look briefly at the software objects it consists of, and how they work together (see Fig. 2.1) [1]:

The *Accompanist* supplies the computer-generated accompaniment of a Blues Band (whose groove etc. can be adjusted using other patterns like the *MetricTransformer* described above). The *HarmonicAnalyzer* uses a root-parsing algorithm as described in standard music literature to determine the current chord (say, Fm^7) in real-time. The *InputAnalyzer* offers a xylophone-like playing metaphor: The user makes downbeat gestures with the two infrared batons of the *WorldBeat* system in his hands. Gesture velocity determines volume, horizontal position determines pitch. The *Corrector* takes this input and maps it to the nearest, harmonically sound note in terms of the current accompaniment chord determined by *HarmonicAnalyzer*.

The ImprovisationHelper has been used already within a computer-assisted improvisation component of the WorldBeat music exhibit. The result is quite fascinating: People who have never before played an instrument, can walk up to the system and start improvising to a Blues band -- without playing wrong notes! This makes this WorldBeat component very attractive and popular among visitors who use the WorldBeat system. Experienced musicians can adjust the support to either a moderately hard to play, chromatic

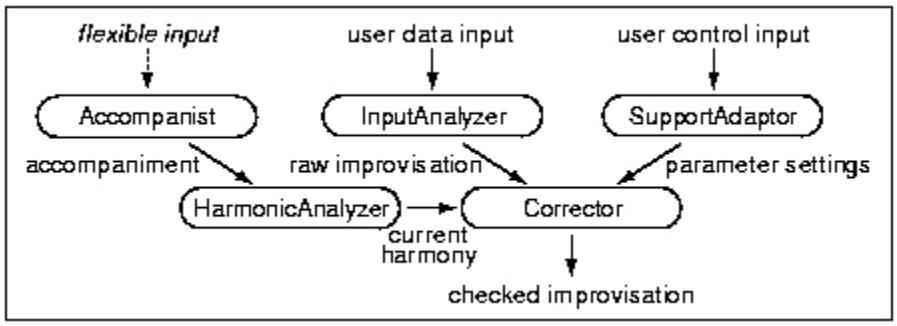

Fig. 2.1: The structure of the "ImprovisationHelper" component.

virtual xylophone using the batons (1 octave), or they can even use a full-scale electronic piano keyboard that can be added to the system at any time (7 octaves).

Interaction and experience design

Supporting rich musical structures also requires an appropriate user interface to ensure that users can actually understand the musical concepts encapsulated in the system, by interacting with them. We have achieved that by using a set of two "infrared batons" that can be used to create musical input. They work like xylophone mallets that are used in mid-air. A tracker converts baton movement to MIDI controller data which is then interpreted and turned into musical events. The system uses the *Lightning II* baton system designed by Buchla and Associates, and adds a software layer on top of this input device that lets the user create musical input for a GlobeMusic application in a very musical way - by downward drumming gestures with the two batons (see Fig. 2.2).

Fig. 2.2: A user improvising to Blues using WorldBeat

These batons offer a number of characteristics that are advantageous to our goal of a broad user base:

They do not look like a standard computer interface, and instead create an innovative appearance that makes the system more tempting to use.

They do not look like a complicated, professional-level musical interface either (such as an 80-key piano keyboard), which would scare off people who do not play such instruments.

They force users to interact with the system in a much more physical way, requiring them to stand and wave their arms to play, instead of just asking the user to sit down passively and operate with icons on a small screen using a mouse and keyboard.

Since there are two batons, *local* cooperation between two players is not only possible, but natural and encouraged. It has been observed already when evaluating the *WorldBeat* exhibit that the two batons often lead people to share them between them, essentially turning the human-computer interaction of a single user creating music into a human-human cooperation where the computer merely becomes a medium. This is an exciting additional level of cooperation in addition to the remote collaboration that we envision with our NetMusic and GlobalMusic systems (see below).

While it is obvious that a special-purpose device such as the infrared batons is not available to every potential user today, we believe that the trend to a larger variety of post-desktop user interface technologies is supporting our vision of a scenario where people will obtain a certain device if it is really tied in to an online experience and promises a new level of interaction.

3 NETMUSIC: COOPERATIVE EXCHANGE OF MUSICAL INFORMATION

Chapters three through five are based on two simple but very far-reaching axioms: i) the margin between two sounds must usually be less than 40 ms for the human ear to grasp them as played at once (cf. [16]); ii) even considering data transfer at the speed of light and unlimited bandwidth, it is impossible to keep the delay of bi-directional transactions under 40 ms around the globe. In this chapter, we describe a basic distributed infrastructure for global

cooperative exchange of musical information, interpreting "cooperative eMusic-making" as (somewhat) asynchronous editing of a common musical score; chapter four emphasizes means for reducing the real time delay between globally distributed musicians to the minimum possible, and chapter five investigates *synchronous* cooperative eMusic-making given that this delay is (typically) still above 40ms.

ResRocket Surfers

A survey [17] considered the approach for musical cooperation used by *ResRocket Surfers* the best one investigated. The underlying idea of this approach was to combine the functionality of a MIDI sequencer with the functionality of a *MUD* (Multi User Dungeon). The users could navigate through virtual music studios, in which they could participate in songwriting. Apart from the possibility to chat with the fellow participants, the users of ResRocket could record MIDI tracks and transmit them to fellow participants. This client-server solution can be classified as an *asynchronous* Internet approach for musical cooperation [10].

In the meantime, *ResRocket Surfers* has become a combination of a chat room system and a transfer system for audio data. The virtual studios are now called sessions. Navigation through them as well as the textual communication within the sessions is based on a Web compliant interface. The ResRocket sequencer has been replaced by an API (Application Programming Interface), which is used to enrich third party software like Steinberg's *Cubase* and Emagic's *Logic* with so-called *Rocket Power*, a possibility to exchange music data over a ResRocket server. At the moment, only ResRocket offers such a server, but Steinberg and Emagic are working on proprietary solutions, too. There are three levels of accounts. The one for participating in public sessions is free, the others which offer rights like creating and participating in private sessions must be purchased.

ResRocket Surfers is designed with professional users in mind: e.g., applications based on "Rocket Power" have complete control over all network events; users are prompted before downloading a session; users can pause data transfer while recording and playback; and users determine when to submit changes [10].

The NetMusic System

The goal of *NetMusic* was to go one step beyond the *WorldBeat* level reported in the previous chapter, towards the networking capabilities envisioned for *GlobeMusic.* Synchronous (real time) cooperation was not yet considered. The concepts for data transfer were conceptually derived from those described for ResRocket Surfers. The user interface and application design philosophy of NetMusic, however, followed the WorldBeat line-of-thoughs – in fact, NetMusic was integrated with WorldBeat and targets both experienced and, in particular, inexperienced users (concerning both musical cooperation and computer usage). From an overly simplistic point of view, NetMusic brings the communication approach of ResRocket to WorldBeat.

Following the principles of simple and intuitive user interface design, most network actions are hidden. As the user clicks the NetMusic button on the main screen of WorldBeat, an automated login and update process starts. The screen changes to the *NetMusic Song Page* (see fig. 3.1). Status messages inform the user about the (usually short) login and update events. Novices receive hints and "how-to"s, users can listen to the (current state of the) song and go further to record a song part i.e. a track (instrument) and take (verse). To prevent fluctuating data during playback, all data are first cached in temporary memory and copied to the main storage when all the data of this part are received. In the standard setup, NetMusic songs are configured to consist of four takes played by four instruments. The choice of instruments serves several requirements: i) predefined instruments simplify the interface for early users, reducing the number of choice options. ii) the instrument pattern of saxophone, piano, bass and drums is often used in modern music and matches well with the sound of typical NetMusic songs, yielding a familiar pattern for most people; iii) the instruments used sound still natural if played with a standard after-touch pattern, making them preferable candidates for the MINI format (replacing standard MIDI) as described in the next chapter. Each part of a song may be in one of the three possible states *free, in progress* and *used*. In the latter two states, songs are marked with the name of the site (e.g., town) where they have been or are edited. This makes the users aware of the distributed nature of *NetMusic*. Unused parts host a free button which leads to the recording pages.

Fig. 3.1: SongPage screenshot (strophe: take)

When the user selects a free part, a lock request is sent to the NetMusic server. The server acknowledges the lock and switches to the recording screen for the desired instrument (except if the lock fails in a raise condition, in which case a corresponding message is displayed). When the user leaves

NetMusic or if the kiosk is reset by the automatic time out mechanism of WorldBeat, the client automatically logs out from the server. A NetMusic server may be co-located with a client and is currently configured to serve up to 16 clients. Its main tasks are client coordination and broadcast, and change logging and propagation.

Asynchronous eMusic-making scheme

The asynchronous scheme described here is to be considered a first step, complemented by a quasi synchronous scheme in GlobeMusic (see chapter 6). It has some roots in both ResRocket Surfer and WorldBeat. There are different ways of recording in NetMusic: for piano and bass, the user plays on a virtual keyboard with the infrared batons. In addition to the improvisation helper function as described in the previous chapter and to the expert mode with full chromatic keyboard, we offer a virtual keyboard featuring all notes of a key such as C-major except for the fourth tone (quart). The drum recording also uses the keyboard metaphor. Several drums are mapped to a simplified virtual keyboard. The user can practice the usage of the instrument first and then start recording by clicking a corresponding button. Saxophone recording follows a special approach: the user hums into a microphone which is attached to the WorldBeat kiosk. The tone hummed is quantized to 12 semitones. In the inexperienced mode, wrong notes are mapped to the best matching ones of the key. The recorded notes are not echoed during recording because differences to the hummed note could disturb the user.

To assist the user regarding timing, a metronome is played in the background during the recording of a take. When the recording starts, the storage contents of the take are copied to a temporary storage in the recorder. This is the storage used locally while the user "edits" i.e. records his take (among others, this prevents simultaneous edits by remote users from disturbing locally). At the end of the recording sequence the take is played once again. Users can listen to their artwork and rerecord it if they are not satisfied. When satisfied with the recording, users navigate back to the song page. The server then broadcasts the new instrument/take to all clients; the clients reflect the update on their song pages.

In NetMusic configurations used for exhibits, a simple exhibition-proof scheme must be used for deciding about the "actual" song edited by globally distributed exhibit clients. The current default scheme maintains a distributed musical score either until it is complete (in the above configuration, 4x4 instrument/takes are recorded and the "last" users at each site have (been) logged out) or when a song has reached 70% completion without remote locks showing up (in this case, it is discarded when the current user logs out).

4 RT-SCORE: QUALITY SCALING OF STREAMING MUSICAL SCORES

The network view: the M-TRTF protocol

It is a well known fact that Internet applications should adapt to the actual condition of the network. This has been proven to be a prerequisite for network stability[1] in [15] and has since been incorporated in the most commonly used Internet transport protocol, TCP. TCP provides a connection oriented service with retransmission and rate adaptation; for most streaming multimedia applications, this behavior is not appropriate. The commonly used alternative, UDP, does not perform any error and congestion control at all and leaves adaptation up to the application. This openness has led to an immense body of research on quality adaptation and end-to-end congestion control as well as diverse applications with a huge range of responsiveness.

Since TCP is the prevailing transport protocol on the Internet, the concept of TCP-friendliness has been introduced for *non* TCP based applications. A TCP-friendly or TCP-compatible flow is defined in [3] as a flow that, in steady-state, uses no more bandwidth than a corresponding TCP flow running under comparable condition. This can be achieved, e.g., by partially emulating the behavior of TCP.

The TCP Congestion Avoidance mechanism divides its sending rate by a factor $\beta = 0.5$ in response to a single loss event and gradually increases it by a step α when no loss is encountered. It is classified as AIMD (additive-increase-multiplicative-decrease); various other AIMD mechanisms, partially using different or varying values for α and β have been proposed for streaming applications [7, 9, 13, 14]. For an adaptive multimedia application, all AIMD mechanisms share the disadvantage of drastically reducing the sending rate in response to a single congestion indication.

Another definition of TCP-friendliness is that a long lived flow should satisfy the TCP response function describing the steady-state sending rate:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t\_RTO * (3\sqrt{\frac{3p}{8}}\, p(1+32p^2)}$$

where T denotes the sending rate in bytes/sec, s is the packet size, R is the round-trip time, p is the steady-state loss event rate and the TCP retransmit timeout value is given by t_RTO [5]. Satisfying this equation actually means sending as much as or less than T, but sending less is not desirable because it may result in starvation when com-

[1] from a distributed application programmer's point of view, the advantage of adaptation lies in a reduced packet loss ratio: it is better to use an adaptable encoding scheme and to reduce bandwidth use in congested networks than to lose a large fraction of data packets

peting with TCP. Using this equation, it is possible to implement a rate-based TCP-friendly congestion control scheme that does not implement AIMD. This scheme called TFRC has in particular been shown to yield a smoother sending rate pattern [5]. For most streaming multimedia applications, drastic bandwidth fluctuations are not desirable since they yield highly jitter-prone streams. This holds true for our application as well. Using the TFRC code from [5], we have compared simulations of TFRC and TCP across a 100 node transit-stub network (which is shown in [18] to be a good approximation of the Internet topology) with a 3:2 mixture of self-similar background traffic (following a Pareto distribution to account for large-scale web traffic [4]) and of TCP flows. It is clearly visible in fig. 4.1 that the sending rate of TFRC is indeed smoother in our simulation scenario. We therefore choose to customize TFRC to our application as M-TRTF (where M stands for music).

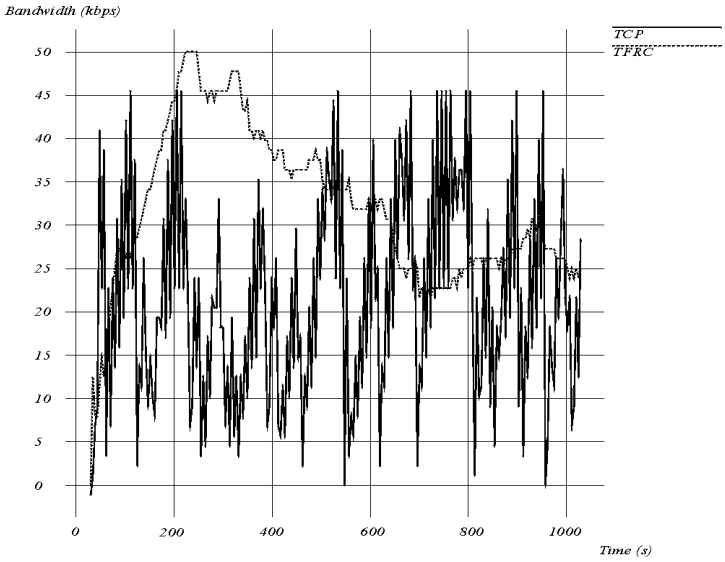

Figure 4.1: TCP vs M-TFRC sending rates.

The application view: the "MINI" musical format

Most research efforts for quality scaling of multimedia data have focused on video or audio. To date, they have hardly dealt with adaptive musical data in a format similar to MIDI, where musical notes are represented by control data (there is an enormous amount of proprietary codes, some of which may contain related mechanisms, but the documentation of such musical codes rarely published or publicly available). This may be due to the fact that many published codes were created at a time when quality adaptation of multimedia traffic on the Internet was not a topic of research. This is true for most codes in [13], which gives a good overview.

Work around structured audio (SA), in particular in the context of the MPEG-4 standard (see [11], [12]), has received a lot of attention in the music community lately. SA aims at combining the richness and fidelity of conventional audio codecs (such as MP3) with the compactness of MIDI. However, SA is design-centered around high fidelity and embraces much more than MIDI devices; as a result, its compactness will usually not go beyond that of MIDI, jitter-prone packet networks are not particularly emphasized, and the encoding / decoding algorithm is processing intensive. MINI, in contrast, has exactly the contrary focus. Note, however, that MIDI is just a format proposal while SA represents a huge multi-party research effort embracing format, codec, and much more. MINI basically suggests adaptations to MIDI that enable quality scaling for appropriate transmission across the Internet.

MIDI was not designed to be transmitted over long distances or packet oriented networks in real-time. The assertion of a maximum transmission distance with a minimum bandwidth of 31.25kbaud (+/- 1%) and a given maximum latency did not raise any need to introduce a special symbol for chords; thus, a chord in MIDI is encoded, transmitted and possibly played back as a series of notes, fast enough to avoid audible side effects [8]. If delay is introduced in between notes, e.g., due to latency, a chord is turned into an arpeggio. Jitter makes this arpeggio sound particularly unpleasant, as we have experienced as part of our trials.

One might question our choice of MIDI as a starting point in the first place. We have chosen MIDI instead of a more flexible standard as a starting point because it is the dominant music encoding standard and it is used for communication with hardware devices in our application – our primary requirement was therefore easy translation between MIDI and MINI in both directions. Our format, MINI ("Musical Instruments Networks Interface") is basically a subset of MIDI with special encoding of chords and notes. It can be used for what is called layered encoding: data are sorted in order of importance and given an appropriate priority. Layered encoding is a prerequisite for quality scaling i.e. the adaptation of required bandwidth (and other Quality-of-Service parameters) to network conditions, which we base on M-TFRC as described.

The derivation of MINI from MIDI can be interpreted as a three-step process as follows.

Step 1: Saving space.

One of the design goals being smaller size, the following MIDI control messages were removed:

- **Tune Request**: This message was only designed for analog synthesizers. Even if such a device was used in a collaborative real-time long distance effort, the separated musicians would not notice the differences in tuning.
- **System Exclusive Messages, EOX** (End of Exclusive): These messages are for vendor-specific features only.
- **Active Sensing Message:** This message is related to permanent tones (MIDI errors) and must be sent at least once every 300ms, which is inappropriate for transmission over a packet oriented network.

- **System Reset, Local ON/OFF, All Notes OFF:** Similar to the Active Sensing Message, these messages are related to MIDI errors and testing. Such problems can and must be solved locally in Internet settings.
- **Omni ON/OFF, MONO ON/Poly OFF, MONO OFF/Poly ON:** MIDI supports logical splitting of communication into channels. For MINI, communication is structured based on the distributed application configuration, so that these commands are filtered out, too. Note that the distinction between transmission of monophonic and polyphonic information is implicitly encoded in our chord representation described below.
- **Song Select, Song-Position-Pointer, Start, Stop, Continue:** These messages are used to control sequencers. They are not relevant in our context.

Step 2: Supporting Jitter-prone networks

In MIDI, one must explicitly disable each tone played with a *Note OFF* message. This approach is not only a huge waste of space for tones which have a fixed length (e.g. a vibraphone, a piano or percussion instruments in the "regular" way of playing - special playing techniques are usually encoded as individual instruments), it is in particular infeasible for jitter-prone networks. In MINI, we use a single bit for identification: if the bit is cleared, the instrument is to be played with a fixed length. This predefined note-length period may be used to define a quasi metronome based playing scheme and may be combined with multiples (e.g., describing quarter, triplet, and half notes if the fixed length defines eighths) and with pauses – the receiving process may then buffer notes and play smoothly and accurately even in presence of considerable jitter. For compatibility reasons, we also support the note-on/off concept (in which case the above-mentioned bit is set for note-on and cleared for note-off); this mode should of course be used over connection-oriented networks or in case of very low jitter only.

Step 3: Layered (chord) encoding

There is no special symbol for chords in MIDI. Apart from the aforementioned arpeggio effect, this is clearly a waste of space. We also assume that the MIDI range of ten octaves will usually not be entirely used for streaming music. According to step two above, we have seven bits left in a byte (one is reserved for NOTE ON / NOTE OFF control information). To determine what we can encode with these seven bits, we recall the formula for the collection M of all sets (here: chords) with at most k elements (here: tones) as a k^{th} order combination of n elements with repetition and without ordering:

$$M = \binom{n+k-1}{k} = \frac{(n+k-1)!}{k!(n-1)!}$$

Using a seven bit word, it is possible to describe all combinations of n and k where M does not exceed 128. This yields the following possibilities:

Encoding	Range (n)	Voices (k)	Combinations (M)
1	15	2	120
2	128	1	128

Extending the concept to two bytes (15 bit), we can look at combinations where M does not exceed 32768:

Encoding	Range (n)	Voices (k)	Combinations (M)
1	12	7	31824
2	14	6	27132
3	18	5	26334
4	28	4	31465
5	57	3	32509
6	255	2	32640

Tables for three, four and five bytes can be found in [17]. Even ambitious keyboard players should be fully satisfied with the possible range of 63 tones and chords of up to ten notes given by a five byte encoding: most semi-professional and professional keyboards have a range of five octaves (60 tones). The five byte encoding is therefore be considered the maximum in the current version of MINI.

Quality scaling is possible in MINI in several variations: i) switching between chord encodings with more or less bytes is supported, with the choice of limiting the tonal range or the number of voices; ii) fine-tuning MIDI messages such as "Program Change" and "After Touch" can be sorted in the order of importance for layered encoding, to be selected for inclusion or omission for quality scaling; iii) the fine-tuning messages may also be switched from "per-tone" to "per-chord" and "per-sequence", offering another dimension of quality scaling.

Combined with a bandwidth adaptation mechanism with a smooth sending rate (in our case, M-TFRC), the frequency of changes in quality scaling was experienced to be quite tolerable in our applications and studies.

Note that the status of MINI as described represents "work in progress" in the state used for our application. Nevertheless, the authors see sufficient potential for this work to become the outset of a standardization effort.

5 GLOBEMUSIC: ASSEMBLING THE PIECES BASED ON A NEW MUSIC-MAKING SCHEME

Despite all the advancements of the parts introduced up to now (WorldBeat, NetMusic, RT-Score i.e. M-TFRC and MINI), the invariant problem of audible signal delays remains in a global scale application. In this respect, we believe in the need for a new cooperative eMusic-making scheme based on the experiences described in the previous chapter: WorldBeat proved the feasibility and appropriate-

ness of new eMusic-making schemes for the individual, and NetMusic successfully adopted and adapted the cooperative eMusic-making scheme introduced by ResRocket. Note that the latter scheme is i) totally asynchronous and ii) appropriate for experienced musicians only (which is clearly not the target we had set out for application targeting the 4th mass (!) medium).

The Take-Shift Scheme

We will start this paragraph with an analogy: global videoconferencing is usually experienced as a synchronous cooperative application despite considerable delays if pairs of audio and video streams are kept lip-synchronous and if, in a dialogue, the network/system-related delay of a stream carrying a response does not exceed the usual "think-time". In a question-answer dialogue, this constraint is usually maintained: as to timing, the human process "ask-think-reply" is usually not noticeably different from the human/network process "ask-transmit-think-reply-transmit" (the computing part is omitted here for the sake of brevity). In a heated debate, however, speakers are often interrupted while talking; such a dialogue may become noticeably different if not unfeasible in a networked videoconferencing setup since the human/network process may have considerably different timing than the human-only process.

The above analogy leads to a key observation:

> *global-scale cooperation can be experienced as synchronous if the technical setup conserves sufficient fidelity of the streams (cf. lip-synchronous above) and if the scheme introduces human delays larger than (or at least in the order of) network delays.*

In our case, the technical constraints are maintained with the introduction of M-TFRC and MINI. As to the *scheme*, the first version of GlobeMusic implements what we call *take-shift* as described below:

- the scheme slightly resembles a jazz (jam session) like setup; instead of hearing himself and all others, a musician hears himself and a limited number of "others" plus a computer provided "initial" background track.
- the WorldBeat "improvisation helper" metaphor is adapted; depending on the experience of the group members, the "degree of help" may vary; e.g., as described for WorldBeat and NetMusic.
- for a given take, musicians are ordered, to be called #1, #2, etc. in the remainder; the lowest order (denoted as #0) is assigned to the computer provided background track
- musician #1 hears the initial background track and adds his own track with the help of the improvisation helper; thereby, #1 hears #0 and #1
- the result is sent to #2 in real time based on M-TFRC and MINI, played out there with a slight delay (with respect to the *start* at #1, not the *end* like in ResRocket!); #2 adds his own track with "his" level of improvisation helper support, hearing #0, #1, and #2 (i.e. himself).
- the result is sent to #3 who "joins in" a tick of a second or at most a few seconds later, and so on.
- Once a take is fully recorded, it may be audited by all group members or the next take may be created
- For subsequent takes, the ordering is re-arranged, following either a cyclic shift rule, or the desires of the "band", or a random drawing
- Videoconferencing may be added for a more intense group experience; a major adaptation is necessary in comparison to standard videoconferencing: during recording of a take, the usual audio channel is replaced by the MINI channel, and "chord synchrony" between the MINI and video streams (as opposed to lip synchrony) must be maintained – with track #0 defining the reference timing for all MINI tracks and for all video streams.

Fig 5.1 depicts the ensemble of GlobeMusic, including the modules introduced in the previous chapters.

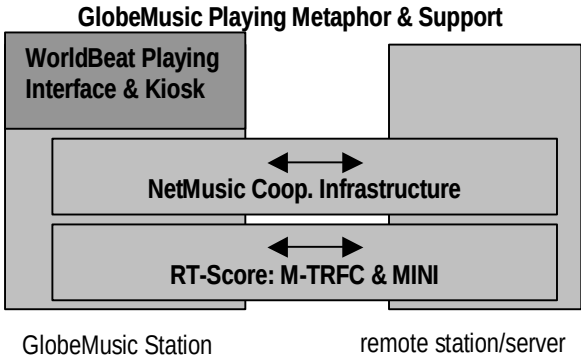

Figure 5.1: GlobeMusic Distributed SW Architecture

6 CONCLUSION

We introduced GlobeMusic as a combination of efforts regarding computer-human interaction, individual eMusic-making, appropriate computer network protocols, musical data formats, and cooperative eMusic-making. Major parts of the system called WorldBeat, NetMusic, and RT-Score were intensively experimented with. WorldBeat has proven the appropriateness of its approaches as an exhibit used by hundreds of thousands of users. Careful evaluation by large user groups and international review boards back this claim further. For the "ensemble" described in chapter 5, such large-scale evaluation is still to be carried out, yet the overall results described in this paper are considered valuable for the WEDELMUSIC audience.

7 REFERENCES

1. Borchers, J. A Pattern Approach to Interaction Design. John Wiley & Sons, March 2001.
2. Borchers, J. WorldBeat: Designing a baton-based interface for an interactive music exhibit. in *Proc. CHI'97, ACM press, 1997.*
3. Braden, B., Clark, D., Crowcroft, J., et al. Recommendations on Queue Management and Congestion Avoidance in the Internet, *RFC 2309*, April 1998.
4. Crovella, M., Bestavros, A. Self-similarity in World Wide Web traffic: evidence and possible causes, *IEEE/ACM Trans. Networking 5, 6 (Dec. 1997), pp. 835 – 846*
5. Floyd, S. Handley, M. Padhye, J., Widmer, J. Equation-Based Congestion Control for Unicast Applications, in *Proc. SIGCOMM 2000.*
6. Geirland, J., Sonesh-Kedar, E. *Digital Babylon: How the Geeks, the Suits, and the Ponytails Fought to Bring Hollywood to the Internet* Arcade Publ., Sept. 1999
7. Jacobs, S. Eleftheriadis, A. Providing Video Services over Networks without Quality of Service Guarantees, *World Wide Web Consortium Workshop on Real-Time Multimedia and the Web*, 1996.
8. Mc.Queer, B. The USENET MIDI Primer, WWW site http://www.harmony-central.com/MIDI/Doc/primer.txt
9. Rejaie, R. Handley, M., Estrin, D. An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet, in Proc. INFOCOM 99, 1999.
10. ResRocket Surfers, WWW site http://www.resrocket.com/
11. Scheirer, E., Vercoe, B. SAOL: The MPEG-4 Structured Audio Orchestra Language.*Computer Music Journal* 23, 2 (Summer 1999), pp 31-51.
12. Scheirer, E. Structured Audio and Effects Processing in the MPEG-4 Multimedia Standard. *Multimedia Systems* 7, 1 (Jan 1999), pp 11-22.
13. Selfridge-Field, E. ed., *Beyond MIDI, MIT Press, Cambridge Mass. etc. 1997.*
14. Sisalem, D., Schulzrinne, H., The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme, in *Proc. NOSSDAV'98, 1998.*
15. Van Jacobson, "Congestion Avoidance and Control", in Proc. SIGCOMM 1988, 314-329.
16. Webers, J. Tonstudiotechnik. Poing: Franzis, 1994
17. Welzl, M. *NetMusic: Real time concepts and systems for telecooperative exchange of musical informatik (in German). Diploma Thesis, Univerity of Linz: Telecooperation*, 1997.
18. Zugra, E., Calvert, K., Bhattacharjee, S., How to Model an Internetwork, *Proc.IEEE Infocom '96*, 1996, San Francisco, CA.