Netjack - Remote music collaboration with electronic sequencers

by Alexander Carôt and Torben Hohn
1.) Introduction

2.) Theory of remote musical interaction

3.) Alternative approaches

4.) Remote music collaboration with electronic sequencers
   (Netjack)
1.) Introduction
The current state:

- In the old days: analogue signal processing
- PC has revolutionized the domain of audio engineering
- HD recording / editing / mastering
- Prof. software e.g.: Ardour, Cubase, ProTools etc.
- MP3, Ogg Vorbis Files → sharing / exchange

Real musical interplay?

1) Introduction
2) Theory of remote musical interaction
2) Theory

- Cognitive Science: Integration Processes
- Ira J. Hirsh: Threshold for order identification = 20 ms
- Slightly higher for interactive music: Schuett/Chafe EPT = 30 ms
2) Theory
2) Theory

EDAL (Ensemble Delay Acceptance Threshold)
2) Theory
Realistic Interaction Approach (RIA)

2) Theory
2) Theory

Speed of sound : 343 m/s
2) Theory

~ 8.5 m

- Soundgenerator
- Air
- Ear
- Brain

- significant delay
- slight delay
- no delay
speed of light: 300.000 km/s (factor ~ 900.000 !)
2) Theory

~7500 km
2) Theory

**Propagation delays – fiber**

- Anti-aliasing filter
- Sampling
- Quantization
- Input filtering
- Input blocking & driver buffering
- Transmission
- Output blocking & driver buffering
- Output filtering
- Reconstruction

28 ms

20 ms
2) Theory
2) Theory
3.) Alternative Approaches
3) Alternative Approaches

Strings Apart
Pedro Rebelo, Mark Applebaum
SARC / CCRMA (8141 km / ~ 100 ms)

Latency Accepting Approach (LAA)
3) Alternative Approaches

Master/Slave Approach (MSA)

Bass-Master-Groove

Master-Side-Sound

Slave-Side-Sound

delay > 25 ms
Laid-Back Approach (LBA)

3) Alternative Approaches
3) Alternative Approaches

Single delayed feedback (SDF)

Dual delayed feedback (DDF)

Delayed Feedback Approach (DFA)
• **EDAL = 0 ms (!)**
• **Approach: Application of SDF (single delayed feedback)**
4.) Interconnection of remote electronic sequencers (Netjack)
History of Netjack1

• Implemented as a patch to jack in 2005.
• Committed to jack-svn 2008-03-06
• In November 2009 I got aware of CELT and added support for that.
• Then started making netjack more robust against packet loss.
• Reimplemented the algorithm used in alsa_io
Synchronise jack-transport

• No vari-speed in jack.
• Only possibility is synchronising sample-clock
• Thats the idea behind netjack.
• Jack-driver which syncs one instance of jackd to another one.
4) Netjack
Codecs

• Mainly 2 types of codecs.
  – High delay (mp3, vorbis, aac)
    • Need >100ms of audio to emit a compressed byte.
    • Suitable for music.
  – Speech Codecs
    • Speex, gsm....
    • Low latency
    • Not suitable for music.

• Need low-latency music codec
CELT

- Closes gap between vorbis and speex.
- xiph.org (BSD style)
- Nice music quality.
- Support for Packet Loss Concealment
- Many different latencies, and bitrates supported.
Low-latency Codec comparison

![Bar chart showing PEAQ scores for different codecs and bitrates.](chart.png)
Bit rate reduced, go internet?

• With the reduced required bitrate it is theoretically possible, to create a netjack link over internet.

• OHNOES. where iz packet?

• A packet may either be late or get lost on the internet.

• Netjack was ignoring this fact, because it doesn't happen on LAN.
How to decide if packet is lost?

• Easy on the master. It expects a packet with a specific sequence number in process_callback().

• On the slave i am using a deadline, which is constantly calibrated. So that a „reply“ to lost packet is still reaching the master in time.
so... netjack1 scales now.

- Zero latency mode.
- 1-3 periods of Roundtrip on LAN
- Arbitrary latency for an internet link.
Why not Netjack2?

• „Jack1 is dead, stop hacking it.“ © sampo_v2
• Netjack2 was written while i was pretty inactive. No backport to jack1 was done.
• Its more comfortable, but purely targeted at LAN. (broadcast)
• Adding CELT support to netjack1 took 1 hour. It was my code.
• The packet assembly code needed only small mod, to become a jitter-buffer.
Both protocols waste bandwidth

- **Netjack1** has a huge packet header
  - 48 bytes per packet. (most of it has redundant data)
- **Netjack2** uses a second sync packet.
  - Also a lot of overhead when encapsulated.
- so... i think, we want netjack3.
Conclusion

• We can use netjack over Internet.
• Needs a nice frontend to setup connections.
• We have a big mess with 2 implementations of jack and 2 implementations of netjack.
• It looks like the mess will get bigger (netjack1 for jack2 is in the pipe)
• Hopefully netjack3 will not make it even worse.