音楽心理学実験ツールとしてのPC環境性能の再検討

長嶋洋一[†1]

概要：過去に多くの機器/システムの遅延(レイテンシ)とばらつき(ジッタ)について計測実験を行い、音楽心理学実験の領域で便利/高性能な道具として普及したテクノロジーを正しく理解せずに使用することは、研究の基盤そのものを無意味にしてしまうと警鐘を鳴らした。そこからPCの性能が約100倍ほど向上し、多種の周辺環境が整備されてきたが、情報技術/情報科学の本質としてのレイテンシ/ジッタは依然として存在している。そこでネットワーク環境や物理コンピューティングの進展を受けて、改めて当時と同様の問題点に関する計測実験を行ってみたので報告する。いくつかの音源システムの発音遅延については別報告に譲り、本稿では計測実験の基盤部分の詳細とともにOSC(OpenSoundControl)の活用について検討した部分を詳しく報告するとともに、逆にレイテンシ/ジッタを積極的に活用する可能性についても検討してみたい。

Re-examination of PC-based environment as a tool of experiment in musical psychology

YOICHI NAGASHIMA[†1]

1. はじめに

約20年前に多くの機器/システムの遅延(レイテンシ)とばらつき(ジッタ)について計測実験を行い、音楽心理学実験の領域で便利/高性能な「道具」として普及したテクノロジーを正しく理解せずに使用することは、研究の基盤そのものを無意味にしてしまうと警鐘を鳴らした[1-5]。そこからPCの性能が約100倍ほど向上し、多種の周辺環境が整備されてきたが、情報技術/情報科学の本質としてのレイテンシ/ジッタは依然として存在している。そこでネットワーク環境や物理コンピューティングの進展を受けて、改めて当時と同様の問題点に関する計測実験を行ってみたので報告する。実験の詳細はWebで公開しており[6-7]、いくつかの音源の発音遅延に関する検討は日本音楽知覚認知学会2019年度春季研究発表会[8]の場で詳細に報告するので、本稿では主として今回の計測実験の基礎部分(実験環境)に関する技術的な検討、およびOSC(OpenSoundControl)の活用に関する検討について報告する。

過去の実験/警鐘の発端となった30年前の研究報告[9]によれば、人間が同時に鳴る3和音を聴取する際に、5msecとか10msecとかのオーダの発音遅延を特定の音に作用させただけで、和音全体の感覚的な印象が異なったという。ここに例えば発音遅延(レイテンシ)の絶対値が20msecのオーダでばらつく(ジッタ)MIDI音源を使用した場合、PCソフト上での時間的データが実験の意味を喪失させるオーダでばらつくのに、ソフトや市販音源を盲目的に使用している研究者がこの事実に気付かない可能性がある。これらの発表後に筆者のところに多くの専門家やメーカ技術者が青い顔をして自分のシステムの性能について問い合わせてきたのは印象的な「事件」だった。

2. 実験環境の概要

今回の実験で使用した計測実験システムの概要について述べる。全景は図1のようなものであるが、先に簡単に二言でまとめると「Max7から115200bpsのシリアルでArduinoのポートを叩いて、Arduinoから2chのディジタル出力を上げ下げしてPicoScope3205Aで計測する。時間差(遅れ)を計測するために、PicoScope3205Aはワンショット・トリガモードで立ち下がりエッジに反応する」という事になる。

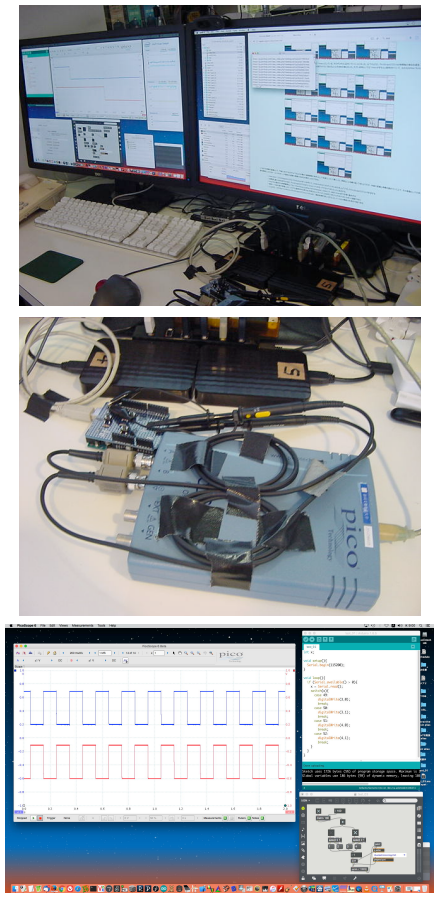

図1 実験システムの全景.

†1 静岡文化芸術大学 Shizuoka University of Art and Culture.

筆者の研究室にあるWindowsパソコンは古風(Windows98と3台のWindowsXP)なので必然的にMac(10数台在籍)ベースで実験を進めた。実験に使用した5台のコンピュータは全てMacOSX 10.11.6 (El Capitan)のMacである。実験を行うメインMac mini(2.6GHz Intel Core i5, 8GB 1600MHz DDR3)およびOSCメッセージを返すサブiMac(Retina 5K, 3.2GHz Intel Core i5, 16GB 1867MHz DDR3)と、3台のMacBookAir(1.4GHz Intel Core i5, 4GB 1600MHz DDR3)(2.2GHz Intel Core i7, 8GB 1600MHz DDR3)(1.3GHz Intel Core i5, 8GB 1600MHz DDR3)を使用した。インストールされているMaxはMax7.3.6とMax8.0.2であり、時間的性能を最高速度(Overdrive,scheduler=1msec,slot=1msec)に設定したのはもちろんである。

PicoScope3205Aとは英国Pico Technology社の2チャンネル100MHzサンプリングオシロスコープで、8ビット精度で500Msamples/sのサンプリングレートで16Msamplesのバッファメモリを持つ。過去には対応PCソフトがWindows版しか出ていなかったが、2018年8月にリリースされたMacOS対応のPicoScope 6.13.7 Beta for macOSという対応ソフトの動作を確認して使用した。なおUSBシリアルドライバについてはFTDIUSBSerialDriver_v2.4.2からバージョンを下げてFTDIUSBSerialDriver_v2.2.18を使用している。この事情についてはWeb公開資料[7]の「USBシリアルドライバについて(補遺)」を参照されたい。これはMacを実験/研究に使用する研究者に必須の注意点として有名である。

時間計測ツールとしてのPicoScope3205Aの性能確認を行った様子を図2に示した[7]。上下2段(青いAチャンネルと赤いBチャンネル)はオフセット設定で上下にずらし、アナログ入力電圧(+5V)がアッテネータで0.5Vとなる。

図2 PicoScope3205Aの実験画面例.

オシロのトリガ設定をワンショットの立ち下がりエッジとし、HとLの中間の300mVをトリガレベルと設定してrunさせて負論理のスイッチを押すと、青いAチャンネルの電圧がトリガレベルになった瞬間が時間軸のゼロとして、それ以前とそれ以後の電圧がサンプリングされて表示される。図2の左上(1目盛り100μsec)では単に+5Vの電圧がGNDレベルに下がるだけと見えるが、右上(1目盛り10μsec)や2段目(1目盛り1μsecと200nsec)になると立ち下がりの瞬間にGNDレベルよりもマイナス側までバウンドする。3段目(1目盛り50nsecと20nsec)とさらに時間分解能をあげていくと、数回のバウンドの最初には-2V以下まで振れ、遠いディジタル出力端子(D2)でも1V近く誘導ジャンプする。図2の最下段(1目盛り10nsecと5nsec)では入力のHからLへの電圧変化はおよそ10nsec近くをかけて偏移する。今回の「音楽心理学実験ツールとして」の用途、そして後述するArduinoを用いたシステム性能(10μsecオーダ)からすれば、計測システムとしてこの性能は十分に高速・正確である事から、本実験の計測ツールとしてPicoScope3205Aを使用することは妥当であると結論づけた。

3. Arduinoの時間的性能確認

使用したのはArduino UNOであり、CPUのクロックは16MHzである。Arduinoスケッチは以下のように「スイッチが押されたら(変化したら)ディジタル出力ポートの信号を反転させる」という最高速の無限ループであり、スイッチOFFの初期値として入力の青いAチャンネルは「H」レベル、出力の赤いBチャンネルは「L」にしておいた。

```
int x, y, z;

void setup() {
    z = 0;
    digitalWrite(2, z);
    y = analogRead(A0) / 512;
}

void loop() {
    x = analogRead(A0) / 512;
    if (x != y){
        y = x;
        z = !z;
        digitalWrite(2, z);
    }
}
```

負論理のスイッチの立ち下がりエッジを検出判定する(H/Lと比較する冗長な)処理が不要である理由は、スイッチの状態変化のたびにディジタル出力が反転すれば、スイッチONで青いAチャンネルは立ち下がりが起きて赤いBチャンネルはL信号がH信号に立ち上がり、スイッチOFFの際にはそれぞれ反転して元に戻るからである。一定の処理時間を要するアナログ入力監視の無限ループのどの瞬間にスイッチが押されるかはランダムなので、Arduinoのクロック速度に対応して一定の時間幅が生じる(ジッタ)[7]。

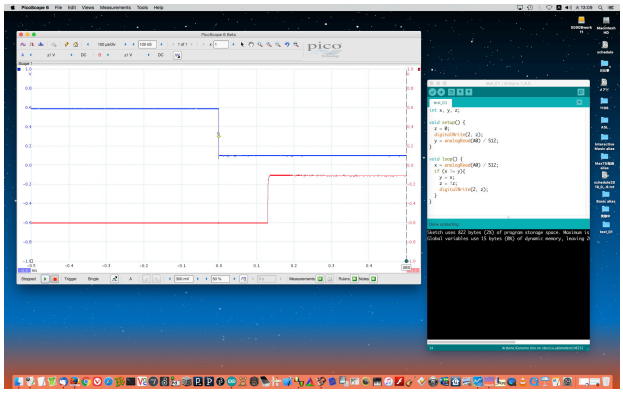

図3 Arduinoの速度実験例.

図3は上記のArduinoスケッチで性能を見るために連続10回、試してみたグラフの一例である(全データは[7]を参照)。およその目分量でこの10回試行の計測データを

130,180,160,150,125,170,150,120,155,205

と読み、平均の遅延は155μsec、最小値が120μsec、最大値が205μsecとなった。最小値と最大値に約2倍近い開きがあるのは、無限ループのどの瞬間にスイッチが押されるかに依存するので当然であり、100μsec程度の誤差がここでのArduinoの実力値であると言える。

次に、上の実験のArduinoスケッチにたった1行、変化したディジタル出力を書き出す直前に「delay(1)」、つまり「1msecのwait」で足踏みするという実験を行った(詳細な全データは[7]を参照)。およその目分量でこの10回試行の計測データを

1.21,1.14,1.13,1.20,1.15,1.18,1.17,1.14,1.17,1.12

と読み、平均の遅延は1.16msec=1160μsecとなった。上の実験の平均値155μsecと比較してもオーダとしていい感じに対応しており、結論として、Arduinoの時間管理が10μsecのオーダでほぼ正確であると確認した。

4. [Max+Arduino]連携システムの時間的性能確認

心理学実験システムとしてMaxからArduinoを経由してPicoScope3205Aで時間計測を行う、というのが基本方針なので、次にMaxとArduinoの連携について検証した。前節の実験ではArduinoシールド上のプッシュスイッチをArduinoがアナログ入力でモニタしてディジタルビットから出力したが、今後はホストがMaxとなって基準のスタート情報もMaxから来るので、処理時間のかかるアナログ入力は不要となり、PicoScope3205Aに接続する2チャンネルの信号をいずれもArduinoのディジタル出力(計2ビット)として個別に出力した。

MaxとArduinoの連携の手法としてはFirmata(+Maxuino)、Arduino2Max、Arduino-USBMIDIの3種類がある[10]が、Arduino UNOではArduino-USBMIDIが使えない。そしてFirmata(+Maxuino)は多機能のため冗長なので、基本戦略としてはArduino2Maxのインターフェース手法に準じることにした。以下はArduinoスケッチであり、115200bpsのシリアルUSB入力があるまで無限ループで待機している。入力が"1"であればPicoScope3205AのAチャンネルに接続されたディジタルポートをLにし、入力が"2"であればPicoScope3205AのAチャンネルに接続されたディジタルポートをHにし、入力が"3"であればPicoScope3205AのBチャンネルに接続されたディジタルポートをLにし、入力が"4"であればPicoScope3205AのBチャンネルに接続されたディジタルポートをHにする 。

```
int x;

void setup(){
      Serial.begin(115200);
}

void loop(){
      if (Serial.available() > 0){
            x = Serial.read();
            switch(x){
                  case 49:
                        digitalWrite(3,0);
                        break;
                  case 50:
                        digitalWrite(3,1);
                        break;
                  case 51:
                        digitalWrite(4,0);
                        break;
                  case 52:
                        digitalWrite(4,1);
                        break;
            }
      }
}
```

以下の図4はホストのMaxパッチの例であり、"1"、"2"、"3"、"4"という4種類のいずれかのデータを115200bpsのシリアルUSBから出力する。スペースキーを叩くと"2"と"4"が送られて、2ビット(→PicoScope3205Aに接続する2チャンネルの信号)ともHとなる(初期状態)。パッチ中央の大きなボタンが基準のスタートであり、メッセージ"1"を送ってAチャンネルの信号を立ち下げる。このパッチ例ではそこから「delay」で設定された時間だけ遅れてメッセージ"3"を送ってBチャンネルの信号を立ち下げる。

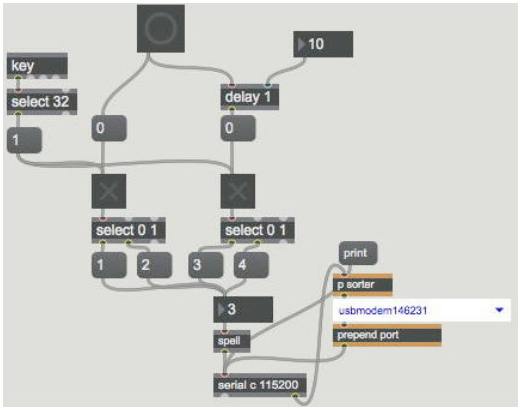

図4 Maxパッチの例.

以下の図5はスタートから10msecおよび1msecの遅延時間を計測した実験の一例であり、複数回の試行からジッタがほぼ無く正確に計測されていることを確認した。なお、最高性能とするために、Max7はOverDriveをONにしていて、Prefernces でスケジューラの精度を最高の1msecにした。以上の実験から、このシステムは「msecオーダ」のレイテンシ/ジッタの計測実験として使える、と判断した。

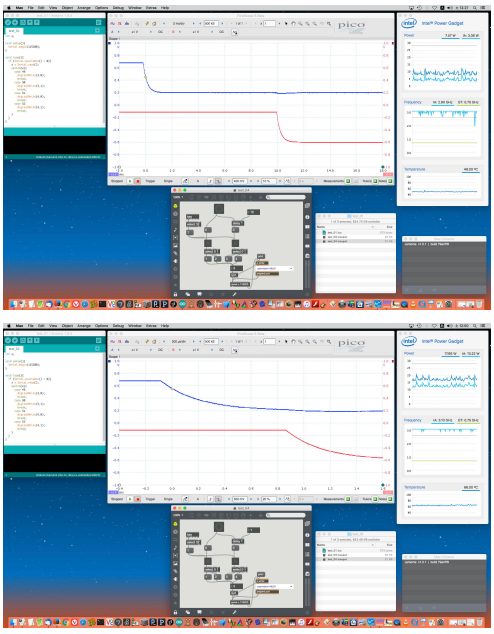

図5 Arduino+Max実験の例(10msecと1msec).

5. OSCの基本能力テスト

マルチプラットフォーム・プロトコルとして世界標準になっているOSC(OpenSoundControl)[11]について、実験システムへの活用性を検討するために、OSC伝送に関するレイテンシとジッタを計測した。「Arduino+PicoScope3205A」のシステム部分はここまでの説明と完全に共通である。実験としては、SUAC北棟11階の1106研究室Mac(Mac mini)からSUAC学内ネットワーク(LAN)を介してSUAC南棟東端4階のマルチメディア室Mac(iMac)との間(約100メートル)でOSCメッセージ”0”を往復させてレイテンシを計測し、複数回の試行によってジッタの傾向を調べた。なお実験は平日の朝の始業前(07:30～08:50)の時間帯で、SUAC学内ネットワーク(LAN)の使用率はほぼゼロに近い環境で実施した。図6は実験に使用したMaxパッチ(1往復)の例であり、上がOSC送信→受信の遅延を計測するメインMaxパッチ、下がOSCメッセージを即時に返送するサブMaxパッチである。

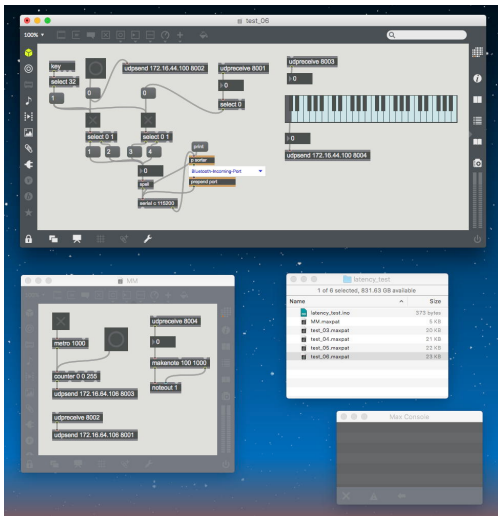

図6 OSC基本実験のMaxパッチ例.

実験はメインMaxパッチ側でOSCの往復回数を「1回」「10回」「100回」として行った[7]。OSCで1往復の実験のレイテンシは約1.0msecであり、複数回の試行でばらつき(ジッタ)がほぼ無いことを確認した。OSCで10往復の実験のレイテンシは10～12msecあたり、OSCで100往復の実験のレイテンシは118～125msecあたりの成績となってジッタが小さいことを示した。これでLANを活用したOSCというプロトコルが高速で良好な性能を持っていることを確認できた。

6. 複数ノートPCのWiFiを経由したOSC

前節のOSC実験では、いずれもLAN(有線)接続のデスクトップPCを使用したが、最近ではノートPCの性能も向上しているので、WiFi接続した複数のノートPCを介在させたOSC伝送について計測実験した[7]。実験は1106研究室内で行い、SUAC学内ネットワーク(LAN)に接続した1106研究室のメインMac(“mini”と呼ぶ)とともに、同じSUAC学内ネットワーク(LAN)に接続した3台のAirMac(WiFiルータ)を介して3台のMacBookAir(ノートMac)もDHCP接続した(ここでは3台のMacBookAirを”Air3”、”Air4”、”Air5”と呼ぶ)。全てのMacでMaxパッチが走り、メインの”mini”は上の実験1と同様に1バイトデータ"0"を"Air3"にOSCで送ったタイミングから、最終的に"Air5"からOSCで戻ってくるまでのレイテンシを計測した。"Air3"のMaxパッチは"mini"から受けたOSCメッセージを即時に"Air4"にOSC転送し、"Air4"のMaxパッチは"Air3"から受けたOSCメッセージを即時に"Air5"にOSC転送し、"Air5"のMaxパッチは"Air4"から受けたOSCメッセージを即時に"mini"にOSC転送する。図7は実験の風景と結果画面の一例である。3台を経由して一巡するのに計3回のOSCメッセージ伝送が行われているが、何回か試行してみるとレイテンシは合計4.5msecほどでほぼ一定していた。折り返しのマルチメディア室iMac G5の「3.2GHz i5」、メインMac Miniの「2.6GHz i5」に対して、ここで使用した3台のMacBookAir3-5は「1.4GHz i5」・「2.2GHz i7」・「1.3GHz i5」とややクロックが低いので、「1106→マルチメディア室→1106」を1msecで行き来するOSCが、WiFi経由のMacBookAirで1台あたり1.5msecで行き来するというのは妥当なオーダである。この実験から、OSCを使うのにLAN(有線)でもWiFi(無線)でも十分な能力があると判断した。

図7 WiFi利用のOSC基本実験の風景と画面例.

7. OSCメッセージを大きくした実験

上記の実験では、OSCメッセージとして1バイトの数値を使用していたが、筋電センサMyoでも脳波センサMuseでもOSCメッセージとして刻々と情報はかなりのボリュームがある[12]ので、ここで図8のように、OSCメッセージをネットニュースから拾った「592文字、107ワードの英文」として実験してみた[7]。OSCはより大きなデータパケット単位で処理されるためか、結果に違いがないことを確認した。これは高度なシステムを構築するのに朗報である。

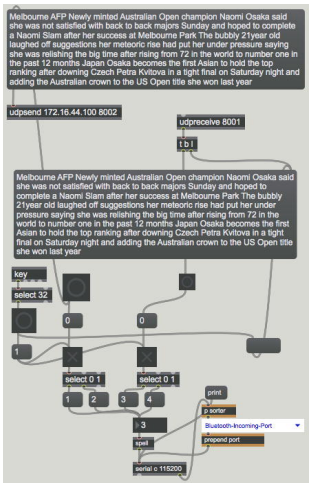

図8 大きなデータのOSC実験Maxパッチ例.

8. ネットワークを混雑させたOSC実験

上述のOSC伝送計測実験の際に、実験に使用してMaxが走っているPCのバックグラウンドで(裏で並行して)動作するブラウザが多数の大容量動画データをアドオンによってダウンロードしていると、OSC往復のレイテンシが100倍程度のオーダで著しく悪化した。この予備実験を受けて、実験PCがOSCメッセージを行き来させている研究室ネットワークにWiFi接続した複数の別のPCが、それぞれブラウザで多数の大容量動画データをダウンロードする環境(ネットワークの混雑がOSCのレイテンシを悪化させるか?)という実験を行った[7]。

まず、ネットワーク負荷がかかる前の状態の実験としてSUAC北棟11階の1106研究室Mac miniをLANに、中継用のMacBookAir3をAirMacでWiFi接続して、SUAC南棟東端4階のマルチメディア室iMacとの間で、上述の実験と同じ592文字/107ワードのOSCメッセージを、「mini→(OSC)→MM室iMac→(OSC)→MacBookAir3→(OSC)→iMac→(OSC)→mini」と2往復させる実験を行った。その結果は図9のようにおよそ8msecで2往復というレイテンシであり、昼間なのでやや増えたジッタとしても「7msecから12msecあたり」でまずまず安定という状態であった。

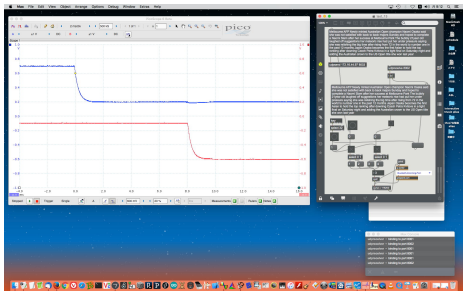

図9 ネットワークの空いている時の画面例.

次に、このレイテンシを計測しているMac mini自体はそのまま最小限のタスク(他にブラウザ等を起動しない)状態で、OSC中継のMacBookAir3がWiFi接続しているWiFiルータのAirMac(5)とは異なる、別の2つのWiFIルータとして新たにAirMac(3)とAirMac(7)をLANに繋いで1106研究室内にWiFiを同時に飛ばした。そしてこの2つの別のWiFiルータにそれぞれ接続したMacBookAir4とMacBookAir5が、それぞれYouTubeで長大な動画として有名な5時間以上の車窓動画を2本ずつ同時並行ダウンロードをする(トラフィック値はSUACネットの上限に近い合計1.5MB/秒ほど)という「ネットワークがかなり混んでいる」状況を作った。YouTube動画は全て最高品質(1920×1080HD)で、さらに異なるコンテンツとしてSUACのプロキシサーバでキャッシュ共用化されるのを避けた。なお違法ダウンロードを避けるために、実験の途中で上の全てのダウンロード作業を強制終了させており、手元にはダウンロードファイルは生成されていない(経過キャッシュはダウンロード中止とともに自動消滅して生成ファイルも消える)。図10はこのネットワークを重くしている2台の画面の例である。その結果は図11のように、OSC伝送一巡のレイテンシは非常に稀な30msecから130msecまで、かなりの幅で最善値の8msecに対して、大きくレイテンシが増大しているジッタを確認した。

ここで確認の追試として、ネットワークを重くしていた2台のMacBookAirを停止させ、実験用Mac miniで同時に(並行して)ブラウザFirefoxを起動して、上の実験と同様の「長大なYouTube動画4本を全て最高品質(1920×1080HD)で同時並行ダウンロードをする」(トラフィック値は合計1MB/秒ほど)という実験を行った。注目点としては図12の画面のIntel Power Gadgetの最下段のCPU Tempetatureの表示が40℃程度のままであり、Activity MonitorのCPU使用率もそこそこ低いことである。このMac miniは4コアのi5でありCPU使用率は最大で合計400%まで行けるので、この数値はCPUはほとんど遊んでいるがネットワークトラフィックがとても重いという条件となる。その結果は、レイテンシが130msecほどの時もあれば190msecほどになっている場合もあり、他の2台のコンピュータによってネットワークが重いという条件よりもさらに悪化した。すなわち、高性能PCだからと言って「1台で実験の全部を行う(システムを分業させない)」事にはリスクがあると考えられる。

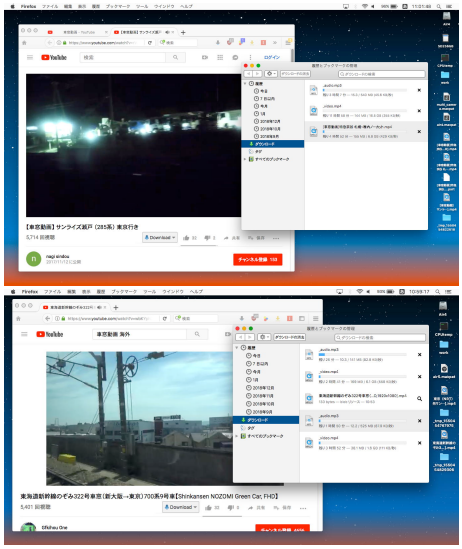

図10 ネットワークを重くした2台の画面例.

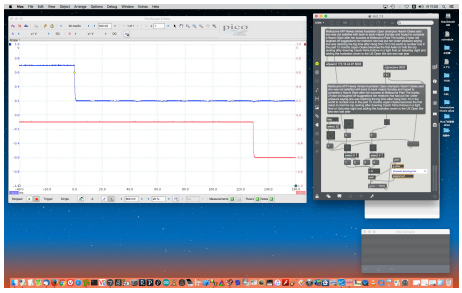

図11 レイテンシが大幅に増えた画面例.

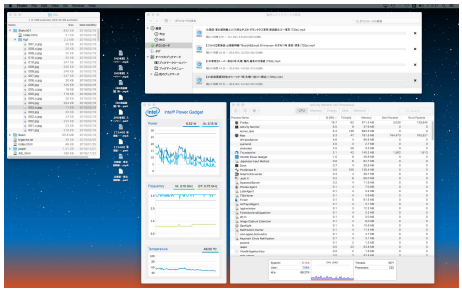

図12 CPU温度CPU占有率の画面例.

次に、上の実験ではFirefoxのダウンロード・アドオンを使用したが、最近ではストリーミング動画中継視聴も多いので、上の実験と同一の環境(1台のMacBookAirのWiFiを挟んでOSC2往復)において、この計測実験を行っているMac miniで同時に並行してFirefoxとChromeとSafariとVivaldiという4種類のブラウザを同時に開き、それぞれ全てabema.tvの同じライヴ中継(藤井7段の将棋)をストリーミング視聴してみた。図13のスクリーンのように4種類のブ

ラウザが同時に起動されている(音声をOFFにしないと、各ブラウザによってストリーミング表示する遅延が異なり、同じライヴ中継なのにバラバラな音響が聞こえて来る)。スクリーン右下のActivity Monitorを見ると、一般に軽いと言われるVivaldiは、その裏でVivaldi Helperというプロセスを走らせてCPUを占有していた(同様のプロセスはChromeにもあり)。この実験の結果、レイテンシとしては10msec以下の時もあれば90msecほどになっている場合もあった。Webストリーミング視聴の場合には「PCが忙しい時には映像が止まってもOK」というベストエフォート原理が強い上に、基本的に「細切れ」でデータが届くようである。1106研究室とマルチメディア室とでOSCパケットを2往復させるという仕事がほとんど「邪魔無し」で走ることもあれば、それぞれのストリーミングでネットワークが重くなり、さらにCPUもブラウザのHelperがデコード処理をそこそこ頑張っている(CPU温度は70℃あたりに上がっている)ので、このような成績となったと考えられる。

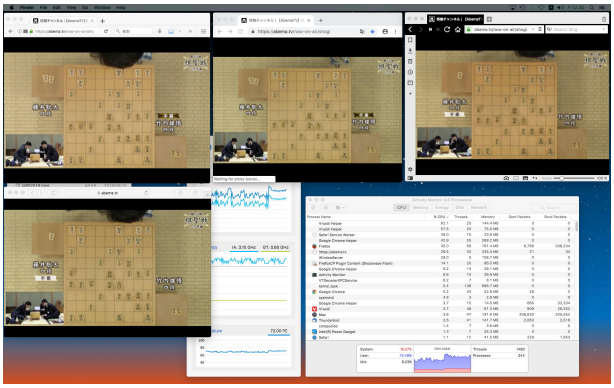

図13 4種類のブラウザでの同時ライヴ視聴の画面例.

9. CPU負荷を重くする実験

上記で登場したIntel Power GadgetおよびActivity Monitorという2種類のユーティリティは、CPU温度とCPU占有率というデータとして刻々と「PCにとって重いCPU処理」を確認するのに役立つ。そこで次の実験として、経験的にCPU処理が重い2種類の動画エンコーダ/コンバータに多数の処理プロセスを同時並行的に投げ込んでみることにした。素材の動画としてエンコーディングに有利な冗長性(背景が単一色とかあまり動かない風景など)があっては困るので、まず図14のような簡単なMaxパッチで640*480ピクセル(30fps)の「画像もサウンドもホワイトノイズ」という1分ほどのmovieを書き出した。これをQuickTime7Proでたくさんコピペ連結して20分ほど続くmovieを作り、mp4-basicフォーマットでExportした結果、ファイルサイズは613.9MBと無駄に大きい(ある程度大きくないと処理が終わってしまうので必要)素材ムービーが完成した。

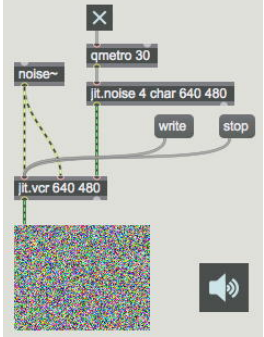

図14 ホワイトノイズmovie生成用Maxパッチ.

CPU負荷を重くした実験の第一は、Free MP4 Converterというフリーのエンコーダ/コンバータソフトである。前節の最後の実験(OSCで2往復計測実験の裏で4種類のブラウザを同時に開いてそれぞれストリーミング視聴)と他の実験条件は同一である。まず最初に、巨大な「画像も音響もホワイトノイズが20分続くだけ」movieをたくさん複製・リネームしてnoise01.mp4からnoise08.mp4まで8本の別々のファイルとした。図14は、これら8本のムービーをFree MP4 Converterによって、レイテンシ計測実験の裏で同時に一気に並列変換している様子である。Intel Power Gadgetのグラフではcpu温度は85℃ほどまで上がっている(外部からファンで冷却)。さらにActivity Monitorを見ると8本のレンダリングHelperが走っていて、それぞれのCPU専有率を合計すると300%以上になっている。しかしCPUは超忙しいもののOSCネットワーク実験の遅延はびくともせず、何度計測してもほぼ8msec(影響ナシ)となった。

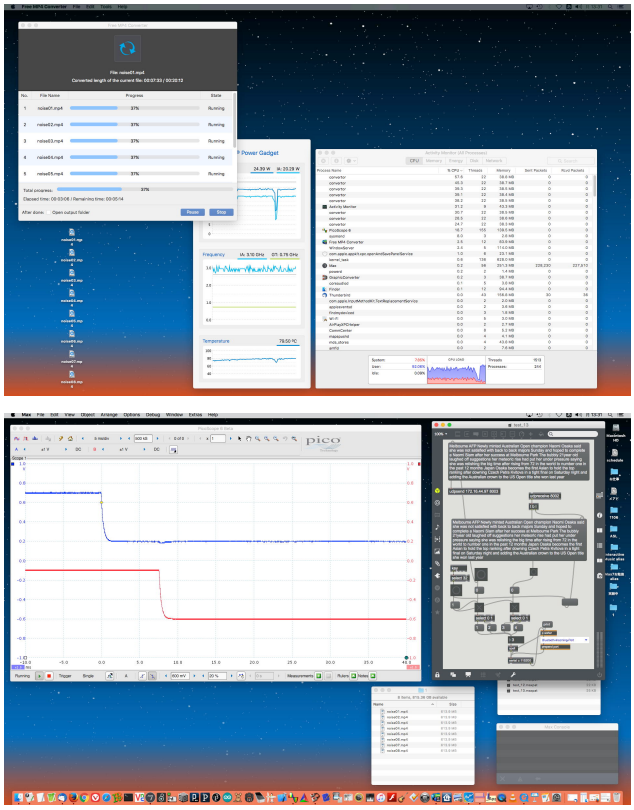

図14 Free MP4 Converterでの実験画面例.

続いてCPU負荷を重くした実験の第二は、QuickTime7Proで同じ8本のmp4-basicフォーマットのmp4ムービーをH.264フォーマットのmp4ムービーに変換する、という同時並行レンダリングである。このmp4エンコーダ・レンダリングは経験的には他のアプリケーションやOS(Finder)の処理を著しくスローにしてしまう事が多い、激重プロセスの横綱である。図15はこの様子であり、CPU温度は最高の90℃以上にまでになったので外部ファンの空冷が必須である。裏で起動された8本のQuickTime HelperのCPU専有率の合計も400%に近いという凄いことになった。しかし結果的にはネットワーク遅延はこちらの実験でもびくともせず、何度も計測したのだが、いつもほぼ8msecとなった。この2つの実験から、いくらCPUは忙しくてもHelperはその合間をスケジューリングして動作するらしく、OSC遅延計測実験の結果には影響しないという結論となった。

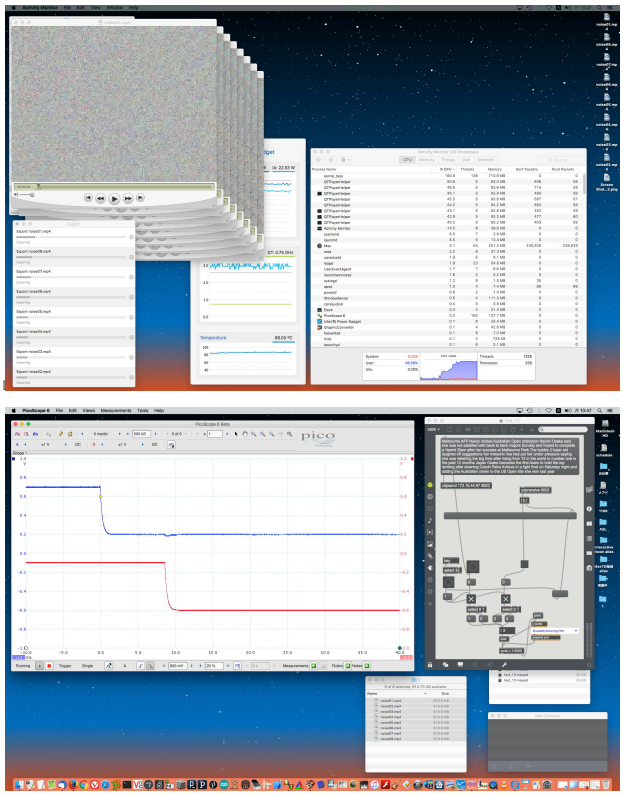

図15 QuickTimeProでの実験画面例.

10. レイテンシ/ジッタを活用する可能性

今回、レイテンシとジッタについて温故知新的に再検討を開始したのには別の理由もある。今後に向けてのアイデア段階であるが、一般的に悪しき/避けるべきものとされるレイテンシとジッタに対して、積極的な活用の可能性を追求していきたいのである。これは、プリゴジン・ストロガッツ・ベルタランフィ・ギブソンなどがそれぞれ提起している新しい科学的パラダイムに対応した視点である。

従来の基礎心理学・実験心理学のアプローチは「被験者へのメディア刺激(五感のいずれか)とそれに対する反応」という図式で構築されてきたが、ギブソンの提唱する生態学的心理学(アフォーダンス心理学)はこの基本を否定して、周囲環境の全体とマルチモーダルにインタラクションする事自体が動物(ヒトを含む)の知覚である、と主張した。実際に生態学的アプローチでなければ説明/検証できない実験心理学的事例(マイクロスリップやアクティブタッチ等)が次々に報告されて、心理学の領域は21世紀になり劇的な変革期を迎えている。そしてレイテンシとジッタはこの生態学的知覚認知研究において重要な意味を持つ、というのが筆者の感触である。

一例を挙げれば、HMDなどVR環境での「3D酔い」の原因の一つは高性能なマルチメディア・コンピューティングの成果として「スムースにヌルヌル動く3次元映像」にあるとされるが、この対策として画質を低下させるという現状は本末転倒であり、解決策の一つとして積極的に「制御されたジッタを加える」という可能性を検証してみたい。ここに関係してくるのが、プリゴジンが道筋を示した複雑系科学/カオスのアプローチ、あるいはストロガッツが提唱した「同期/共鳴」の科学(本質的にレイテンシとカオスを考慮する事になる)、さらにベルタランフィの一般システム論の考え方による取り組みである。

また、ここ数年のトレンドとして、深層学習AIとIoT(どこでもネット)を結び付けるキーテクノロジーのリザバーコンピューティング(Reservoir Computing)において、非線形現象(同期/共鳴/カオス)が重要な意味を持つと報告され、関連して、本質的な情報処理時間のばらつき(ジッタ)特性に対して積極的に「カオスの淵」(筆者も1990年代前半にカオス音楽研究の論文で愛用したフレーズ)の付近を導入しようと提案する報告に驚いたところである。

時間学的には、レイテンシは定量的な事象、そしてジッタは定性的な(カオスに至る数理科学的)事象、という性格があるとも言え、悪者とされたレイテンシとジッタを使いこなすというまったく新規な「逆転の発想」を追求していきたい。

11. おわりに

主としてレイテンシ/ジッタ計測実験環境について、さらにOSCを活用したシステムについての実験に関して報告した。同時期に行った「Maxで鳴らすいろいろな音源」の発音遅延については、日本音楽知覚認知学会2019年度春季研究発表会での報告[6][8]を参照されたい。音楽情報科学において「時間」は永遠のテーマであり、まだまだ実験してみたい事項が浮かんでくるので、また次に機会があれば続報してみたい。

参考文献

1. 長嶋洋一. ハード音源／ソフト音源のMIDI発音遅延と音楽心理学実験環境における問題点の検討, 平成11年度前期全国大会講演論文集2, 情報処理学会, 1999 http://nagasm.org/ASL/ipsj1999/index.html
2. 長嶋洋一. MIDI音源の発音遅延と音源アルゴリズムに関する検討, 情報処理学会研究報告 Vol.99,No.68 (99-MUS-31), 情報処理学会, 1999 http://nagasm.org/ASL/paper/ss1999.pdf
3. 長嶋洋一. MIDI音源の発音遅延と音楽心理学実験への影響, 日本音響学会音楽音響研究会資料 Vol.18,No.5, 日本音響学会, 1999 http://nagasm.org/ASL/paper/onchi99.pdf
4. 長嶋洋一. 生体センサによるパフォーマンスとシステムの遅延/レスポンスについて, 平成14年度前期全国大会講演論文集4, 情報処理学会, 2002 http://nagasm.org/ASL/ipsj2002/IPSJ0203.pdf
5. Yoichi Nagashima. Measurement of Latency in Interactive Multimedia Art, Proceedings of International Conference on New Interfaces for Musical Expression, NIME, 2004 http://nagasm.org/ASL/paper/NIME04.pdf
6. http://nagasm.org/ASL/Latency_Jitter/index.html
7. http://nagasm.org/ASL/Latency_Jitter/test.html
8. 長嶋洋一. PC環境での心理学実験におけるレイテンシとジッタの再検証, 日本音楽知覚認知学会2019年度春季研究発表会資料, 日本音楽知覚認知学会, 2019 http://nagasm.org/ASL/paper/onchi201906.pdf
9. 梅田利彦, 山田真司, 北村音一, ピアノの長3和音の各音の発生時間による変化, 日本音響学会講演論文集, pp.447-448, 1989
10. http://nagasm.org/ASL/Sketching/
11. http://opensoundcontrol.org/
12. Yoichi Nagashima. Bio-Sensing and Bio-Feedback Instruments --- DoubleMyo, MuseOSC and MRTI2015 —, Proceedings of 2016 International Computer Music Conference, ICMA, 2016 http://nagasm.org/ASL/paper/ICMC2016_nagasm.pdf