A System for Precise End-to-End Delay Measurements in Video Communication

Christoph Bachhuber, Eckehard Steinbach
Technische Universität München
Chair for Media Technology
christoph.bachhuber@tum.de, eckehard.steinbach@tum.de

Abstract

Low delay video transmission is becoming increasingly important. Delay critical, video enabled applications range from teleoperation scenarios such as controlling drones or telesurgery to autonomous control through computer vision algorithms applied on real-time video. To judge the quality of the video transmission in such a system, it is important to be able to precisely measure the end-to-end (E2E) delay of the transmitted video. We present a low-complexity system that automatically takes pairwise independent measurements of E2E delay. The precision can be far below the millisecond order, mainly limited by the sampling rate of the measurement system. In our implementation, we achieve a precision of 0.5 milliseconds with a sampling rate of 2kHz.

1. Introduction

With the advent of 5G networks and the prospects of the tactile internet, researchers such as Fettweis [2] suggest an E2E delay of 1 millisecond for communication systems of the future. This shall enable applications such as closed feedback loops for fast assembly robots, highly dynamic teleoperation in virtual or augmented reality, car-to-X communication to improve safety and efficiency in transport and many more.

For all of these, video transmission can be used as a low delay information carrier. Therefore, we need a precise measurement of the E2E delay of video transmission systems. These measurements are preferably non-intrusive, such that they can be applied to a wide range of systems. Furthermore, a video camera typically has a fixed refresh rate, producing new images in constant time intervals. Real world events are virtually never synchronized to these camera frame intervals, also called frame periods. To make a correct measurement, we have to trigger a real-world event at a random time. Because of this, we will never have a deterministic E2E delay, but obtain statistical distributions for a series of measurements.

Many approaches to measure partial delays are well known, for example to measure the processing delay on a camera or the encoding latency. For both the signal propagation time through the circuit has to be measured, which is a standard task in hardware design. But there are few approaches available to measure the end-to-end latency of the more complex system of an entire video transmission chain. This measurement also comprises delays from data transmission between processing blocks and the synchronization effects between blocks operating at fixed rate.

1.1. Related Work

Several methods to measure E2E delay in video transmission have previously been proposed. An overview of their system characteristics is given in Table 1.

The implementations in [3, 6] rely on local measurements. This means the presentation of a clock, for example on a computer screen. This clock is filmed, the image of it transmitted and displayed by the video transmission system under test. Another camera films both the real clock and the clock displayed by the video transmission system. By comparing the clock states in the resulting image, the E2E delay can be computed. These methods suffer from many issues: without image processing algorithms, the calculation of the delay has to be done manually by reading the numbers from the final image. For the measurement system, one has to purchase an additional camera to record the entire scene. Further, it is very imprecise because the clock and second camera are refreshed at a comparably low frequency, e.g. 60 Hz for state-of-the-art systems. The phase shifts between all the sampling elements like cameras and displays add uncertainty to the measurement.

Imagine two cameras filming a virtual digital clock on a display, all three running at a frame rate of 60Hz. Let us for the sake of the simplicity of this example also assume that the exposure time of a camera is half a frame period. One camera is perfectly in phase with the display, meaning that
whenever a new frame on the display is shown, the camera starts exposing. And the other camera is out of phase such that it starts exposing half a period (8.3 milliseconds) after the image on the display has been updated. In this case, the first camera will make the same image available 8.3ms earlier than the second one. The analog insight holds true for the proposed measurement system. The issue is that we don’t know the phase of the cameras and displays, so the measurements will not be precise. This is a fundamental issue with many delay measurement methods that we tackled with our implementation.

Jacobs et al. [4] set the basis for our system: the authors use a blinking light-emitting diode (LED) in the field of view of the camera as signal generator and tape a photoelectric sensor to where the LED is shown on the display. The LED triggers an oscilloscope which also records the signals from the photoelectric sensor. This allows them to manually extract the E2E delay of individual samples. The problem is that this method is not automated on a simple circuitry and therefore requires high effort and expensive equipment. Moreover, they do not offer a thorough analysis of the measurements since absolute E2E delay measurement is only one part of the bigger paper.

Sielhorst et al. [7] propose a system that comprises moving LEDs. From the position difference of the LEDs in the actual world and on the video, the delay is automatically computed by employing a computer vision algorithm. This method does not include the exposure delay of the camera since the source continuously creates events (new translation positions). So the recorded positions always match the frame exposure. Furthermore, they use a recording rate of the measurement camera of at most 200 Hz. This introduces an imprecision of 5 milliseconds or even more for slower systems.

Boyaci et al. [1] measure the capture-to-display latency between a caller and a callee in a video conferencing application. They embed timing information in the form of an EAN-8 barcode in the recorded frames. This information is decoded on the callee PC and compared to the internal clock in software. The method is constrained to desktop computers, since it is intrusive and requires custom software to be executed on the caller and callee machines. The authors assume synchronized clocks and take no further analyses or measures to ensure synchronization. Finally, the method does not include the delay introduced by the graphics buffer and the display, since the timestamp is compared to the current time immediately after decoding.

Jansen et al. [5] utilize QR codes to mark time. A measurement system feeds QR codes from a display to the camera of the system under test, from which the video is displayed and again recorded by the measurement system. The measurement system decodes the QR code and computes the E2E delay. The problem is as previously mentioned that a camera is not a time-precise recording tool. With a refresh rate of 30Hz, only the phase shift between the display of the system under test and the measurement camera accounts for 33.3 milliseconds of measurement imprecision. Further, a computer or laptop and a camera have to be used as measurement system, which constitutes one of the most expensive options here.

### 1.2. Contribution

We propose an E2E delay measurement system that unifies most of the benefits of the existing systems as shown in Table 1. It is an advancement of Jacobs’ [4] system and comprises an LED as light source and a phototransistor (PT) as light detector. The actual LED can cover only a small area of the video image to not bias the coding process. The analysis of the data is not done manually with an oscilloscope, but automatically with a microcontroller board. To enable this, we implemented a filter - detector structure as presented in Subsection 2.3. Further, we devised a simple decorrelation mechanism decorrelating subsequent delay measurements to retrieve meaningful data. We propose a framework to compute sampling delay and analyze first

| Author            | Automatic | Non-Intrusive | Decorrelated | Cost  | Precision | Usable by laymen |
|-------------------|-----------|---------------|--------------|-------|-----------|------------------|
| Hill/MacCormik [3, 6] | no        | yes           | no           | medium | low       | no               |
| Jacobs [4]        | no        | yes           | no           | medium | high      | no               |
| Sielhorst [7]     | yes       | yes           | no           | medium | low       | yes              |
| Boyaci [1]        | yes       | no            | no           | none   | low       | yes              |
| Jansen [5]        | yes       | yes           | no           | high   | low       | yes              |
| Our method        | yes       | yes           | yes          | low    | high      | yes              |

Table 1: Comparison of delay measurement methods. Justification of the parameters is found in subsection 1.1, our method is presented in section 2.
measurements obtained with the new system.

The remainder of this paper is organized as follows: Section 2 describes the system principle as well as the hardware and software implementation. Section 3 presents and discusses results obtained with the measurement system. Section 4 summarizes the results and gives an outlook to future work in this field.

2. System Description

2.1. Principle

The measurement is based upon the idea that the video transmission system delays the propagation of light, as depicted in Figure 1. An initially disabled light source is put in the field of view of the camera. The camera records images containing a dark light source, those images are processed, transmitted and finally shown on the display. The light source is switched on at time $t_0$ by increasing the voltage $U$ from $U = 0 \text{V}$ to $U = v_1$, as shown in Figure 2a. After that, the camera starts recording images with a bright light source. These images also undergo processing, transmission and display delays. When the display shows the first bright frame, it starts to emit light with higher intensity. A light sink or photoreceptor is placed such that it can detect a change in light intensity from the display. In photoresistors and phototransistors for instance, a spontaneous rise in light intensity decreases their resistance $R$ from $R = R_0$ to $R = R_1$ depicted in Figure 2b. This happens at $t_1$, which has to take place after $t_0$ if the system is causal. Overall, the light source has been switched on at $t_0$ and the first indication of that on the light sink happens at $t_1$, so the E2E delay $T = t_1 - t_0$ is the difference of the times.

The principle assumes an ideal system without any reaction delay within the light source and sink and with no noise. In our implementation in Subsection 2.2, we tried to keep those reaction times as small as possible and to suppress noise we are using a detection algorithm proposed in Section 2.3.2.
2.2. Hardware Realization

To realize the idea proposed in Section 2.1, we created a circuit shown in Figure 3 with an Arduino Uno as control system. The actual prototype is depicted in Figure 4. It can be connected to a PC using USB or to devices like smartphones using Bluetooth.

An LED acts as light source in the field of view of the camera. In LEDs, the time between the start of an electrical current pulse and the start of emission of photons is called turn-on delay. This adds to the optical rise time, which comprises the amount time that is needed to go from 10% to 90% emitted light intensity at full current. The sum of those two for typical LEDs is typically below one microsecond, in the magnitude of nanoseconds. Since our measurements are in the order of milliseconds, the delay from the LED is negligible. The light sink is a phototransistor (PT), which has rise and fall times depending on the manufacturer, but within a similar range across manufacturers. The PT used by us has a rise and fall time of 10 microseconds, which is also small compared to the actual delay we want to be able to measure. The resistance of the PT is indirectly read out using a voltage divider with a resistor as shown in Figure 3. The voltage is sampled between the PT and the resistor at the MSMT-Pin of the control system. So the control system knows when it turned on the LED \( t_0 \) and needs to find the time \( t_1 \) when the resistance \( R \) of the PT dropped. This is discussed in Section 2.3.

2.3. Signal Processing

The voltage versus ground level is sampled at 2kHz at the MSMT pin in our prototype. The resolution of the voltage is 10 bit. To extract \( t_1 \), the sample data undergoes a two-step processing: first, a maximum smoothing filter and second a rising edge detection algorithm are applied. The algorithm has been validated by comparing the resulting E2E delays with manually read values from an oscilloscope which is connected to the LED and PT.

2.3.1 Maximum Filter

The maximum smoothing is required to suppress wrong detections caused by pulse width modulation (PWM) of the display backlight. In modern panels, manufacturers don’t change the voltage of the backlight to adjust the brightness, but use (PWM) with adjustable duty cycles. This leads to sample values as shown in Figure 5a. It can be seen that the width of one PWM cycle is 5 samples. When considering the 2kHz sampling frequency, this corresponds to a 400Hz PWM cycle frequency. As can be seen, the maximum value starts to rise at sample 44, this is where the brightness of the display increases.

A similar pattern with short peaks is observed in CRT and Plasma monitors: they don’t use a backlight but have extremely short light emitting events. These events are much shorter than the frame period, so again the maximum filter has to be used to guarantee correct rising edge detection. The filter has two tasks: smooth the signal from unwanted waves and let the resulting signal increase immediately if the input signal increases. This is solved by the maximum filter with length \( k \). For every new raw sample \( a_i \), the maximum

\[
b_i = \max_{\text{max}(0,i-k) \leq j \leq i} (a_j)
\]

of itself and the previous \( k \) samples is stored in the processed value \( b_i \). The resulting graph with \( k = 5 \) looks as in Figure 5b. Note that there are two main rising edges: The first at samples 1 to 5 and the second at samples 44 to 49. We only want to detect the latter, since the first is just an initialization effect. Therefore, no edge detection is performed over the first \( k \) samples and we make sure to never trigger the LED during recording these samples.
2.3.2 Rising Edge Detection

To automatically find the sample at which a consistent increase of the sample values is initiated, we apply rising edge detection on a set of \( N \) samples as presented in Algorithm 1. The algorithm has to be robust against noise from room lights and panel refresh. For presentation purposes, we assume that all samples are already recorded. In the actual implementation, it is applied while recording samples. This online detection enables our system to be included in larger, more complex measurement systems.

The rising edge detection algorithm 1 is based on slope thresholding. For every sample, the previous slope \( ps \) is assigned and the current slope \( cs \) is computed in lines 3 and 4 of Algorithm 1. If both slopes are positive, the number of successive positive slopes \( cps \) is incremented and the cumulative positive slope \( cups \) is increased by the current slope \( cs \) (lines 6 and 7). If \( cs \) is not positive, both \( cps \) and \( cups \) are reset (lines 10 and 11).

A rising edge is detected if both \( cps \) and \( cups \) exceed their corresponding thresholds \( \text{thr}_{cps} \) and \( \text{thr}_{cups} \) or if the current slope \( cs \) is bigger than \( \text{thr}_{cups} \). The last condition is to deal with very short, but significant sample value increases. This is particularly important for a signal as in Figure 5b, since it has flat areas in between the light pulses from PWM. If this condition, in line 13, is true, the rising edge timestamp \( n \) is set to the current sample index \( i \). We don’t subtract the filter length because when we are currently at sample \( n \), we have a significant increase in brightness on the display, such that we can say that the event is now fully transmitted. From \( n, t_1 \) can be computed depending on the initiation time and the sampling rate.

The parameters \( \text{thr}_{cps} \) and \( \text{thr}_{cups} \) can be used to adjust sensitivity against robustness. A combination of \( \text{thr}_{cps} = 2 \) and \( \text{thr}_{cups} = 7 \) has worked well on state-of-the-art liquid crystal displays. As we saw, these values heavily depend on the sensitivity of the PT, we use the OSRAM LPT80A. Other filter algorithms such as comparing the current sample to the mean \( m \) of the previous \( k \) are also possible. In this case, a rising edge could be detected if the current sample value \( b_i \) is higher than an absolute threshold \( m + \text{thr} \) or higher than a relative threshold \( m \cdot (1 + c) \). These options have not been tested because Algorithm 1 works sufficiently.

![Figure 5: Signals from the MSMT pin](image)

Algorithm 1 Rising Edge Detection

\begin{verbatim}
1: \( i = 1, ps = 0, cs = 0, cps = 0, cups = 0 \)
2: \textbf{while} \( i \leq N \) \textbf{do}
3: \hspace{1em} \( ps = cs \) \hspace{1em} \text{Previous Slope}
4: \hspace{1em} \( cs = b_i - b_{i-1} \) \hspace{1em} \text{Current Slope}
5: \hspace{1em} \textbf{if} \( cs > 0 \) AND \( ps > 0 \) \textbf{then}
6: \hspace{2em} \( cps = cps + 1 \) \hspace{1em} \text{Count pos. Slopes}
7: \hspace{2em} \( cups = cups + cs \) \hspace{1em} \text{Cumulative pos. Slope}
8: \hspace{1em} \textbf{end if}
9: \hspace{1em} \textbf{if} \( cs \leq 0 \) \textbf{then}
10: \hspace{2em} \( cps = 0 \)
11: \hspace{2em} \( cups = 0 \)
12: \hspace{1em} \textbf{end if}
13: \hspace{1em} \textbf{if} \( (cps \geq \text{thr}_{cps} \text{ AND } cups \geq \text{thr}_{cups}) \text{ OR } cs \geq \text{thr}_{cups} \) \textbf{then}
14: \hspace{2em} \( cps=0 \)
15: \hspace{2em} \( cups=0 \)
16: \hspace{2em} \( n = i \) \hspace{1em} \text{First significant sample of rising edge}
17: \hspace{2em} \textbf{return} \( n \)
18: \hspace{1em} \textbf{end if}
19: \hspace{1em} \textbf{end while}
20: \hspace{1em} \textbf{return} \( 0 \)
\end{verbatim}
actly every one second, we will always measure the same
E2E delay. The concept of an event passing through a se-
quence of processes with a fixed sampling period is further
explained in the following Section 2.3.4.

In reality, those processes do not sample at exactly 60Hz,
but a slight deviation of that rate. Therefore, the shift be-
tween the processes changes depending on the rate dif-
ference. With measurements done at constant frequency,
we only look at a few, repeating realizations of this shift,
as in Figure 7. These samples are also heavily corre-
lated. To overcome these constraints, the measurement fre-
cuency in our implementation is not constant, as the inter-
measurement delay is randomly computed after every mea-
surement.

2.3.4 Sampling delay

A sampling process forwards incoming packages at fixed
time periods. In this section, a package is a general de-
scription of the object being forwarded. This can for example
be a number of bytes passing through data processing steps,
a visible event happening in front of a camera or a physi-
cal package such as a screw undergoing various production
steps. An example of a process is a display with a fixed
rate of 60Hz. No matter when the connected graphics unit
provides a package, in this case a raw display image, the
package will be a number of bytes passing through data processing steps, an event (the
view at a fixed rate. Again, if an event (the
rate of 60Hz. No matter when the connected graphics unit
provides a package, in this case a raw display image, the
package will be a number of bytes passing through data processing steps, an event (the
view at a fixed rate. Again, if an event (the
provides a package, in this case a raw display image, the
display samples and updates it every 16.7ms. The analog
holds for a conventional camera: it samples the image in
the field of view at a fixed rate. Again, if an event (the
package”) takes place in this field of view, it is forwarded
not immediately, but with the next sampling.

The time a package has to wait until being forwarded af-
after arriving in a sampling process is called sampling delay.
To explain this, we analyse one process \( i \) of a chain of \( N \)
subsequent processes. These processes pass a package from
process 1 along all other processes to process \( N \). The model
assumes instant propagation of a package: no additional de-
lay for processing or buffering is included.

Process \( i \) starts forwarding incoming packages with an
offset \( s_i \) relative to \( t = 0 \), as in Figure 6a. This means
that if the process has received a package before \( s_i \), it will
propagate it to the next process \( i + 1 \) at time \( s_i \). If a package
arrives at any of the times \( [s_i, s_i + p_i] \), it is propagated at
\( s_i + p_i \). In general, the timespan

\[
P_k = [s_i + (k - 1) \cdot p_i, s_i + k \cdot p_i]
\]

is called period \( P_k \) with period length \( p_i \), where \( p_i \) is the
constant sampling period for process \( i \). If a package arrives
at \( t_i \) during \( P_k \), it leaves the process at the end of the period
\( t = s_i + k \cdot p_i \), depicted in Figure 6b. Given process \( i \)
with initial offset \( s_i \) and period length \( p_i \), we can now compute
the sampling delay \( d_i \) of a package entering at time \( t_i \). First,
we have to find during which period \( P_k \)

\[
\min k \quad \text{s.t.} \quad s_i + k \cdot p_i \geq t_i, \quad k \in \mathbb{N}_0
\]

the package entered the process. With this information,
the sampling delay

\[
d_i = s_i + k \cdot p_i - t_i
\]

can be obtained. Assuming the processes communicate
instantly, the time at which the package arrives at the next
process \( i + 1 \) equals the time at which it leaves process \( i \).
Therefore, we can use

\[
t_{i+1} = t_i + d_i = s_i + k \cdot p_i
\]
as entering time for the next process. With this frame-
work, the E2E sampling delay

\[
D = \sum_{i=1}^{N} d_i
\]

of an arbitrary number of \( N \) subsequent processes can
be computed. The method can easily be extended to include
processing and transmission delays. We will use it to gain
more insights into the E2E delay of video transmission once
we have set up a partial delay measurement.

3. Measurements

3.1 Setup parameters and measurement
metrics

We present some measurements conducted with our pro-
otype described in Section 2. We have two simple system
setups: System A is a Windows computer with a Logitech
QuickCam QuickCam E3500 webcam with 30Hz. The video stream
is directly displayed on the monitor (Fujitsu Scenicview at
59 Hz) using Skype preview. System B is a Fedora 20 PC
with an AlliedVision Guppy PRO F-031C IEEE 1394 cam-
era and a Samsung 2233BW monitor at 60Hz. As software,
we use coriander 2.0.2.

The E2E delay distribution of 1000 measurements of
system A is shown in Figure 8. The delay is at minimum

![Figure 6: Initial offset and sampling process](image-url)
41.2ms, this means that it takes at minimum 41.2ms from an event taking place until it is shown on the display. This can also be thought of as best case measurement. The opposite, the maximum delay is 124.7ms, representing the worst case delay from the event until the display depiction of it. The 95% confidence interval from fitting a normal distribution to the histogram in Figure 8 for the mean ranges from 75.8ms to 77.5ms. The standard deviation is 13.6ms. The triple (minimum delay / mean delay / maximum delay) sufficiently describes the E2E delay characteristics of a system, so this is the final metric we report for a system. The mean is the mean of the fitted normal distribution. For system A, the delay characteristic is (41.2/76.7/124.7)ms.

For system B, we have plotted these values in Figure 9 as a function of the frame rate of the camera. All values are monotonically decreasing with ascending frame rates. This is because the camera sampling delay is limited by the camera frame period and the frame period is the reciprocal of the frame rate. The 95% confidence interval for the mean estimation lies between the curves MeanUpper and MeanLower. The delay distributions of the different frame rates provide no further insight over the distribution in Figure 8 and are therefore not depicted. The E2E delay characteristics for the 25Hz and 307Hz camera frame rates are (46.2/70.4/96.7)ms and (8.7/16.3/23.7)ms, respectively. The standard deviation, called SD in the legend in Figure 9, also decreases as the frame rate increases because of the shrinking difference between maximum and minimum delay.

3.2 Discussion

The histogram of relative frequencies in Figure 8 is similar to a normal distribution. This is because the E2E delay is the sum of uniformly distributed random delays. One of these is for example the exposure delay: the trigger time $t_0$ of the LED is independent of when the exposure starts and ends. If the LED is triggered with the exposure start, the exposure delay is maximum. When the LED is triggered later during the exposure process, the exposure delay decreases accordingly. The analog holds true for the display: it is not synchronized to when an image from the camera is ready for displaying in the PC. These and other sampling processes give the normal distribution of the E2E delay in Figure 8.

It is furthermore interesting that even the minimal delay in Figure 9 is affected by an increasing camera frame rate. This is because the transfer time of a frame to the computer memory equals the frame period in case of the Guppy PRO F-031C, as we found with a partial delay measurement prototype that is not yet ready for publishing. For a frame rate of 25Hz, the transfer time is 40ms. The minimal E2E delay for 25fps in Figure 9 is 46.2ms, so the remaining parts contribute at least 6.2ms and at most 46.7ms, since the maximum delay is 96.7ms.

Note that to those remaining parts, the exposure delay of the camera also counts. For a 25Hz recording rate, the exposure delay is at most 40ms and theoretically at least 0ms. Though it will never be exactly 0ms, because then, the LED would light up an infinitesimal small time before the exposure ends. In this case, not enough light from the LED arrives on the sensor to trigger the PT on the display side. Practically, the minimum delay from exposure is significantly greater than 0ms. We plan to investigate this dead zone with future implementations.

It can be seen that with higher camera frame rates, both the minimum delay as well as the difference between maximum and minimum delay decreases. The former is because of the frame transfer time from the camera, the latter because of the exposure time.

4. Conclusions

We proposed an inexpensive, automatic and highly precise E2E delay measurement system. It unifies advantages of previously proposed implementations and can be integrated in bigger, more complex video transmission systems. Furthermore, we briefly discussed the origins of delay in video transmission.

So far, we lack insight into partial delays to explain the phase shift between sampling processes with evidence from real implementations. This is a task for future implementations. We also want to investigate the size of the dead zone at the end of camera exposure. Finally, to extend the range of applications and enable more than only local measurements, we plan to implement a system to send LED initialization timestamps over a network.
Figure 8: E2E delay measurement distribution for system A

Figure 9: E2E delay measurement distribution characteristics for different frame rates of the camera in system B

References

[1] O. Boyaci, A. Forte, S. A. Baset, and H. Schulzrinne. vdelay: A tool to measure capture-to-display latency and frame rate. In Multimedia, 2009. ISM’09. 11th IEEE International Symposium on, pages 194–200. IEEE, 2009.

[2] G. P. Fettweis. The tactile internet: Applications and challenges. IEEE Vehicular Technology Magazine, 9(1):64–70, 2014.

[3] R. Hill, C. Madden, A. v. d. Hengel, H. Detmold, and A. Dick. Measuring latency for video surveillance systems. In Digital Image Computing: Techniques and Applications, 2009. DICTA’09., pages 89–95. IEEE, 2009.

[4] M. C. Jacobs, M. A. Livingston, et al. Managing latency in complex augmented reality systems. In Proceedings of the 1997 symposium on Interactive 3D graphics, pages 49–ff. ACM, 1997.

[5] J. Jansen and D. C. Bulterman. User-centric video delay measurements. In Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, pages 37–42. ACM, 2013.

[6] J. MacCormick. Video chat with multiple cameras. In Proceedings of the 2013 conference on Computer supported cooperative work companion, pages 195–198. ACM, 2013.

[7] T. Sielhorst, W. Sa, A. Khamene, F. Sauer, and N. Navab. Measurement of absolute latency for video see through augmented reality. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 1–4. IEEE Computer Society, 2007.