AdScorer: An Event-Based System for Near Real-Time Impact Analysis of Television Advertisements (Industry Article)

Pål Evensen* Hein Meling
paal.evensen@altibox.no hein.meling@uis.no
Department of Electrical Engineering and Computer Science
University of Stavanger, Norway

ABSTRACT

The media measurement industry is in turmoil, with the old prediction-based models being challenged by more accurate measurement techniques, based on actual viewer behaviour drawn from much larger sample selections. As measurement methods converge across different types of media, the online/offline measurement divide will diminish. Television is one such medium that has traditionally required offline measurements. Advertisers are, for the most part, still accepting predictions and historical behaviors rather than current facts. Despite the limited accountability, yearly spendings on television advertisements are still much higher than for any other medium, and rising.

In this paper, we present AdScorer, a scoring system for television advertisements. Our system is based on event stream processing techniques, and can compute scores for advertisements in near real-time based on channel change events from viewer set-top boxes. Our results show that AdScorer is capable of delivering detailed scores on a per-advertisement spot basis for a whole block of commercials, immediately after the commercial break has ended. The scores include regional breakdowns with viewer numbers and shares for each geographical region of Norway as well as national scores. Our evaluation of AdScorer demonstrates that it is capable of scoring numerous channels simultaneously. In our experiments, we used one machine to analyze five channels, but our system can easily scale to support hundres of channels by adding more machines.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks—*Distributed systems*

*Pål Evensen is also with Altibox.

DEBS '12, July 16–20, 2012, Berlin, Germany.

General Terms

Experimentation, Measurement, Human Factors

Keywords

TV advertisement viewership scoring, event processing

1. INTRODUCTION

In the traditional mass media marketing model of printed media, radio, and television, up front prediction and subsequently rating the exposure of an advertisement is based on statistics collected from a very small fraction of the total viewership. Thus, significant inaccuracies can be expected in these statistics. Moreover, this inaccuracy mainly exists because of the broadcast nature of the traditional mass media model, in which media consumers are secluded from providing feedback to the broadcaster [10].

However, in recent years we have been shifting away from this traditional model to an Internet-based model in which media consumers are empowered with numerous additional capabilities. With this model, the audience is no longer a passive crowd of media receivers, but increasingly active participants, uploading videos on YouTube, blogging, and interacting with each other on social media platforms such as Twitter and Facebook.

Additionally, the pervasiveness of devices such as set-top boxes (STBs) with recording capabilities, smart phones and tablets has enabled people to create their own daily media schedule, where they can choose what media to consume, where and when. Thus, with this changing media landscape comes new opportunities for more accurate prediction and analysis of audience behavior and responses. However, despite the advantages of online advertisement in terms of accountability and targeting, yearly spendings on traditional television commercials is rising [15].

In previous studies, researchers have collected channel change events from networks of STBs and used the data for offline and even online computation of statistics for a number of parameters, including rating of programs and program categories [6, 9]. The primary focus of this paper is online analysis of the impact of advertisements on channel change behaviors among a large population of viewers. The analysis provides a score for each individual advertisement spot. The resulting scores can be useful for numerous parties, such as TV networks, advertisers, and cable network operators, as well as the general public.

To facilitate online analysis, we have developed AdScorer, which combines numerous advanced technologies, including Event Stream Processing (ESP), video stream content recognition, and message-oriented middleware, in order to generate an instantaneous evaluation for each advertisement spot.

AdScorer is deployed in a network with more than 350.000 STBs distributed amongst 250.000 households in Norway. These are spread across the whole country, and represents more than 11% of Norway's 2.2 million households [18], which is a sufficiently large and diverse sample to be statistically significant. As such, this gives us an excellent opportunity to observe how the system performs in a large-scale, real-world setting.

Several algorithms exists for evaluating television advertisements [8], but to our knowledge, none of these works carries the near real-time aspect of our system, nor have they been deployed in a live IPTV network at any scale. Furthermore, none of these covers the complete value chain necessary to perform such calculations, which include STB clients, channel change event collection, distribution and aggregation layers for translating channel change events into statistics, detection of advertisements from the TV channel stream, and finally provide a score for each advertisement.

In previous work [9], we have demonstrated how viewer statistics can be generated in near real-time from processing STB zap events, both by using a specialized event processing language (EPL) and a general purpose programming language (Java). The system presented here, builds on the previous implementation, in the following ways: It has been extended to score advertisements, and it has been embedded in a generalized ESP architecture, presented in Section 4. Section 3 explains how AdScorer is used to provide a *success score* for each individual advertisement during commercial breaks, and in Section 7 we evaluate our implementation of AdScorer. Section 8 surveys related work, and in Section 9 we speculate about the future of media measurement. Finally, we present our conclusions in Section 10.

2. BACKGROUND

We discuss the current state of the media measurement industry, how it is maintained, and why it needs to change.

2.1 The Current State of Media Measurement

Despite the fact that viewing habits and media delivery methods has changed drastically over the last decade, the basic methodology for measuring the impact of television content is still the same as in the early nineties [13], with some aspects, like the survey part, dating back to the fifties.

While the public's response to web advertisements can be analyzed and evaluated in near real-time with reasonable precision and confidence, advertisers are generally limited to base their evaluation on surveys and the daily logs of a small sample of selected households (7500 in the US as of 2010 [7]) when it comes to advertisements presented on television. A detailed survey of the current state of television audience measurement is provided in [9].

As the traditional method of television content distribution (one-way broadcast) is replaced by full-duplex distribution capabilities made possible by IPTV, the traditional survey approach used to estimate television viewership will be replaced by more accurate methods, by capitalizing on STB data [7, 12, 8, 9]. We argue that there is no longer any reason to base the analysis on surveys, other than it being the established *currency* that the industry knows, referred to as *entrenched practices* [7, 10] by some.

In the 1950s, the Nielsen company [16] invented the rating system that dominates the television industry today. Here in Norway, the main provider of viewership data to the official television networks is TNS Gallup [19], a company that specializes in polls and ratings. The measurement methods of this company is still the same as those pioneered by Nielsen.

The status quo in the media measurement industry is maintained by contracts that make it very difficult for competitors to unseat the existing players. Yearly spendings on long term contracts are estimated to be around $50 million per year [10] between networks such as NBC Universal and Nielsen. And the contracts runs for a long time; the current contract between TNS and the Norwegian networks runs from 2008 until 2015.

In general, we believe that some advertisements may annoy viewers more than other advertisements, which in turn can cause some viewers to change the channel. Thus, negatively affecting the impact of subsequent advertisements within the same commercial break, as well as the rating for the actual program(s) separated by the commercial break. This is a good argument against the current method used in both US and Norwegian television networks, where advertisements are displayed in random order within the commercial breaks [12].

3. SYSTEM ARCHITECTURE

This section gives a high-level overview of AdScorer's system architecture, which consists of the following components:

- Broadcast television network
- STB client software
- Advertisement detector component
- Message-oriented middleware
- EventCaster event processing middleware
- Time-series database

AdScorer is an event-driven architecture, in which event producers and event consumers are decoupled [17]. Figure 1 illustrates how the components interacts at a high level. On the left side of the figure, we have the two main event producers, the AdDetector located in our data center and a large number of STBs located in cable customers homes. The AdDetector component automatically identifies advertisement spots in the television video stream, and subsequently publishes an event containing:

- An identifier for the advertisement
- The channel name
- Start or stop status for the advertisement

These events are published to a message queue, and subsequently picked up by another component, as we explain in more detail later. Additionally, the STB clients generates several event types:

- Channel change event (also called a zap event)
- HDMI connection (TV set) on/off event

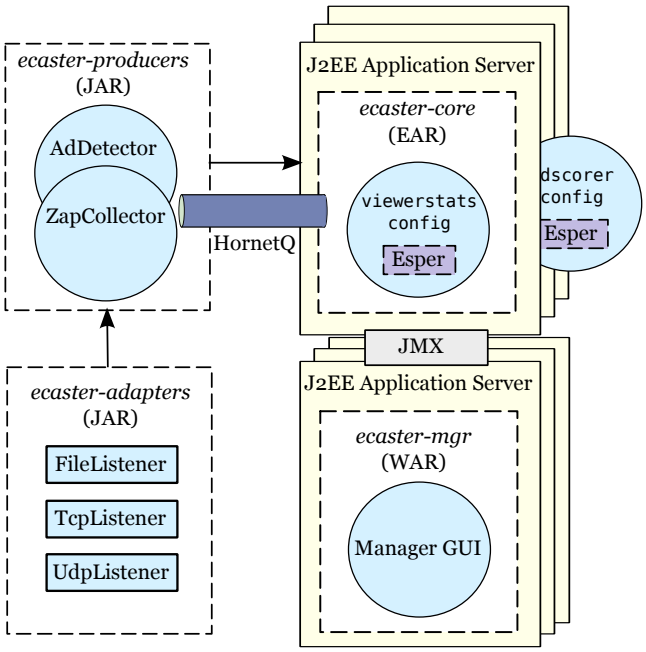

Figure 3: EventCaster Overview

- STB audio on/off event (mute)
- STB volume change event

These STB events are transmitted over UDP to a ZapCollector in our data center. The ZapCollector decodes the packets and places them on a message queue as well.

AdScorer uses two instances of the EventCaster core application, presented in Section 4. One instance continuously aggregates the channel zap events and mute/unmute events into viewer statistics and publishes them on the message queue every second. The other instance subscribes to these *ChannelStat* events as well as *AdStart* events emitted by the AdDetector, as described above. It is responsible for scoring the advertisements, according to the criteria presented in Section 5. Components of the EventCaster middleware are colored yellow in Figure 1.

The sequence diagram in Figure 2 shows the interaction between the components during the evaluation of an advertisement.

4. EVENTCASTER

This section describes the technical details of the EventCaster middleware, and explains the choices of underlying technologies. Written in Java, the EventCaster middleware facilitates general event processing by combining our own Java interfaces with XML, J2EE technologies and the Esper Complex Event Processing (CEP) engine. The Esper engine is programmed using the declarative Event Processing Language (EPL), which builds on the SQL-92 syntax.

Figure 3 illustrates the EventCaster architecture: Software packages are denoted with dotted lines, instances with blue circles, objects with blue rectangles, and application server instances with yellow squares. In the lower left part, we have the *ecaster-adapters*, which are general protocol adapters, used by *ecaster-producers* to receive data. In the AdScorer system, the *ecaster-producers* consists of one ZapCollector-instance and one AdDetector-instance.

The *ecaster-core* package contains the Esper CEP engine, and is deployed on several instances of the JBoss Application Server. All instances are identical, except the main configuration file, which contains: the EPL queries, and defines the connections between specific queues and *event processors*, as well as between EPL statements and *event publishers*. The lower right part of Figure 3 shows the *ecaster-mgr* package that contains a web application for managing the queries and its connections to *event publishers*.

Being a platform for distributed event processing, its main interaction model is publish/subscribe, which allows event producers and consumers to be changed without affecting other parts of the system, and at the same time eliminates some of the latency and processing overhead that typically comes with request/reply style interactions.

Thus, HornetQ was chosen as the distribution layer for events due to its high performance, ease of use, and the fact that it is open source. Events are distributed on multiple queues, according to event type, which allows for greater flexibility and cleaner code on the consumer part. That is, there is no need to set up a filter for extracting different event types. Some may argue against this design, in favor of using only a single queue for all event types, because the temporal ordering of the events might be affected under high load. However, we argue that this is more of a theoretical than practical concern, as the 5 minute average CPU load typically hovers around 5% for the HornetQ server during normal operation. We currently run it on a virtual machine less powerful than your average desktop.

Management of the event processing engine (*ecaster-mgr*), however, is exposed over a request/reply interface, due to the interactive nature of handling configuration changes through a web-based user interface.

Esper was chosen as the event processing engine for much of the same reasons as HornetQ; it is open source, it integrates easily into Java projects, and is easy to set up and configure. Moreover, since Esper rely on EPL for specifying queries, which is based on the SQL-92 syntax, some of the burden of learning a new declarative language is eliminated, provided that the developer already knows SQL.

A J2EE application server comes with useful abstractions for handling things like database connections, access control and application management. HornetQ is built and maintained by the JBoss community, and the JBoss application server comes with HornetQ support out of the box, and as such, was a natural choice of application server.

The use of an application server allows for the convenience of packaging the main package (*ecaster-core*) together with all its dependencies in a single Enterprise Archive (EAR), deployable to the application server. The building and packaging of EARs can be automated using a build tool like Maven [14], and is an elegant way of handling dependencies.

An essential element of the EventCaster middleware is a pair of interfaces, named *EventProcessor* and *EventPublisher* as shown in Listings 5 and 6. These are used to bind various event processors and event publishers to EventCaster and its associated Esper engine and EPL queries. To facilitate this binding, EventCaster requires a configuration file as shown in Listings 1 and 2. This configuration file defines the set of EPL queries to install and which event processors to

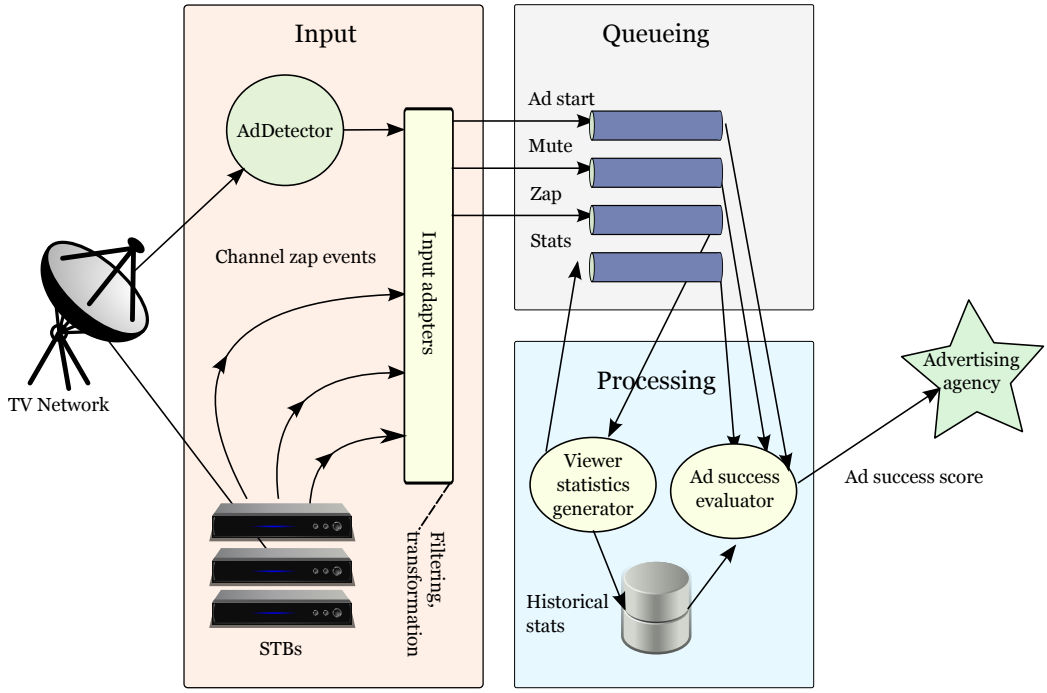

Figure 1: AdScorer Architecture Overview

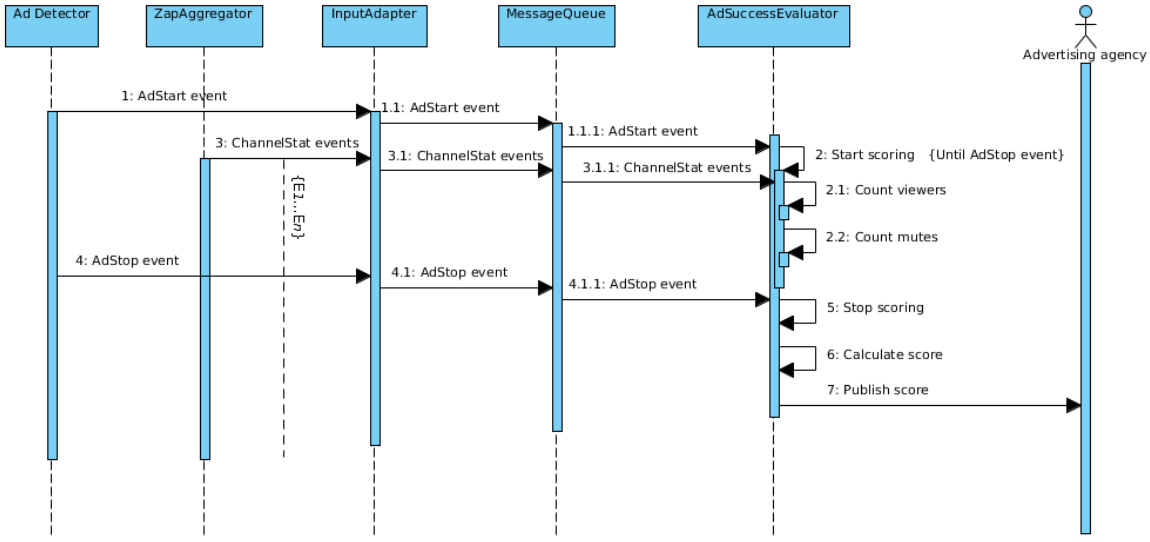

Figure 2: Ad Scoring Sequence Diagram

be loaded at startup. This configuration can be managed in several ways. One approach is to use a Java Management Extensions (JMX) interface accessible through the web interface of the application server, another is to use the JConsole tool bundled with the Java SDK, or the *ecaster-mgr* web interface.

The *EventPublisher* interface is used both by the *input adapters* and for event processing within the middleware. A Java class implementing the *EventPublisher* interface can be connected to EPL queries, and can publish their output events in a variety of ways. Currently, we have implemented publishers that output events to e-mail, text file, databases in addition to the HornetQ message bus. Additional types of publishers could easily be implemented on demand, for instance using HTTP POST as the method of delivery.

Events originate from outside of the system, and are piped onto the message queue by *input adapters*, which are standalone clients for receiving events over various protocols.

These events are then routed to the processing engine by connecting a Java class that implements our *EventProcessor* interface, to a message queue relevant to that event.

Listing 1 is an excerpt from the EventCaster instance doing the advertisement impact evaluation, and illustrates how two *EventProcessor* interfaces are connected to different message queues. Listing 2 is from the configuration of the EventCaster instance for aggregating viewer statistics, and shows how publishers are connected to EPL queries.

Listing 1: EventCaster Processor Configuration

```
<!--List of processors to be loaded at startup-->
<processors>
  <processor>
    <name>tv.ChannelStatProcessor</name>
    <input-resource>lvq.tv.stats.viewerstats</input-resource>
    <enabled>true</enabled>
  </processor>
  <processor>
    <name>tv.AdStartProcessor</name>
    <input-resource>tv.entries.input.ad</input-resource>
    <enabled>true</enabled>
  </processor>
</processors>
```

Listing 2: EventCaster Query Configuration

```
<!-- Decorating the program stats with percentage -->
<epquery>
  <statement>
    @Name('ZapSnapInsert')
    insert into ZapSnap
    select *, percent(viewers, sum(viewers)) as activity
    from ChannelWin
    output snapshot every 1 seconds
    order by viewers desc
  </statement>
  <listeners>
    <listener>
      <name>tv.TvStatSnapshotPublisher</name>
      <output-resource>lvq.tv.stats.viewers</output-resource>
    </listener>
    <listener>
      <name>tv.TvStatDataMiner</name>
      <output-resource>viewer_stats</output-resource>
    </listener>
  </listeners>
</epquery>
```

Listing 3: EPL AdScorer Whole Break Query

```
select * from pattern [
 every
  a=tv.CommBreak(begin=true)
   -> b=tv.AdStat(channel=a.channel)
  until c=tv.CommBreak(begin=false, channel=a.channel)
]
```

The simplicity of the EPL language is shown in Listing 3, which is the query for collecting all *AdStat* events for a television channel between two *CommBreak* events. A listener, implemented in Java, then publishes the result via e-mail. Listing 4 is more complex, and shows the EPL query for collecting and generating statistics on a per-advertisement basis. The *retained* and *iar* functions are custom written Java methods developed specifically for calculating audience retained through the advertisement.

Listing 4: EPL AdScorer Per-ad Query

```
insert into tv.AdStat
select
 a.adId as adId,
 a.channel as channel,
 a.time as startTime,
 b.time as stopTime,
 a.ipList.size() as viewersBegin,
 iar(a.ipList, b.ipList) as iar,
 retained(a.ipList, b.ipList) as retained
from pattern [
 every (
  a=AdIdComplete(begin=true)
   -> b=AdIdComplete(
    begin=false,
    channel=a.channel,
    adId=a.adId)
 )
]
```

Listing 5: Event Processor Interface

```
public interface EventProcessor<T> {
  public void processEvent(T event);
}
```

5. SCORING ADVERTISEMENTS

In our implementation, an advertisement spot is evaluated according a wide range of criteria, as listed in Table 1.

These scoring criteria may be represented in both actual numbers, and additionally in percentage form, in the cases where they are related to other numbers, such as interval between number of viewers at the start and end of the advertisement. The Initial Audience Retained (IAR) criteria, proposed by Dorai et al [8], is the fraction of viewers retained for the duration of an advertisement, and is calculated as follows:

$$IAR = \frac{\epsilon}{\alpha}$$

where ϵ and α is defined in Table 1.

By eliminating the viewers that were not present at the start of the advertisement, the expect to eliminate most channel surfers (viewers that constantly flicks between channels during the commercial break). Combined, these criteria make up a final score, represented as a numerical value between 0 and 10. This score says something about the impact of the advertisement, and gives advertisers an unprecedented opportunity to measure the impact for each individual advertisement spot, based on factual observations, as opposed to claimed attitudes and numbers.

6. DEPLOYMENT

We now describe the current deployment and our planned deployment with enhanced STB client software, which is expected to facilitate significantly more accurate statistics and enable us to conduct more interesting behavioral analysis of television viewers.

6.1 Current Deployment

The STB clients currently deployed in the Altibox network only reports channel change events where the viewer remains on the same channel for more than one minute. In addition, the forwarding of recorded channel change events is delayed until either a 30-minute timeout expires, or a total of ten

Criteria	Symbol
Number of viewers at the start of the advertisement	α
Number of viewers at the end of the advertisement	θ
The interval between α and θ	τ
Number of viewers that stayed on the channel throughout the advertisement	ϵ
Number of viewers that muted the sound during the advertisement	Δ
Number of viewers that was in mute mode when the advertisement began	γ
Number of viewers that unmuted the sound during the advertisement	δ
Number of viewers that turned on TV during the advertisement	Ω
Number of viewers that turned off TV during the advertisement	ω
Initial Audience Retained (ϵ/α)	IAR

Table 1: Scoring Criteria

Listing 6: Event Publisher Interface

```
public interface EventPublisher<T> {
  void publishEvent(T event);
  /**
   * @return address of the published events
   */
  String getDestination();
}
```

channel changes have been collected in a buffer. Unfortunately, this sampling mechanism prevents us from capturing interesting behaviors of television viewers, e.g. the channel surfing behavior during commercial breaks. Additional details regarding the functionality of the current STB client reporting software can be found in [9].

A new STB client software has been developed, in which channel changes and other STB events are forwarded much more rapidly. The new STB client will also forward STB events related to mute, volume, and HDMI status on or off. The latter will allow us to determine if the TV connected to the STB has been turned off. We provide further details on how the new STB client will be used in our planned deployment below.

Unfortunately, it was not possible to deploy the new STB in time to obtain results for this paper, as company policy intending to secure the stability of the IPTV platform means that there are very few windows of opportunity for updating the STB client software during the year. This means that scoring results from the current deployment does not include several of the values from Table 1. The results presented in Section 7 were obtained using data from the old STB client software.

The AdDetector part of the scoring system is commercial software from a vendor that also delivers content-recognition technology to some of the major players in the media measurement industry.

6.2 Planned Deployment

A better understanding of viewer behavior will hopefully be gained when the new STB client software is deployed, as the reduced reporting interval will capture most of the channel surfers, as well as expanding the viewer action repertoire by including mute and volume events.

Furthermore, HDMI status monitoring will address the main criticism against STB-based viewer statistics, namely that most people do not turn off their STB. As such, it is impossible to determine whether there are people watching unless there is STB event activity. Being able to detect whether the TV is turned on or off, enables us to establish with a great deal of confidence whether someone is watching, as virtually everyone turns off the TV when going to bed or leaving the house. It may still be running in the background, while people are doing other things, but then again, the traditional methods are no better in this regard.

It is presently unclear what the impact of this flaw in our previous statistics [9] and other IPTV measurements [6] will be. But we expect it to be significant, as we discuss next.

The new STB client has been successfully tested in Altibox' lab, and was recently deployed to customers as a silent upgrade, which means that only those that power cycle their STB device will be upgraded. Those that leave the STB on, will be upgraded later during a forced upgrade. One day after deploying the new STB software, approximately 15.000 STBs had upgraded, and after a week 80.000 STBs had upgraded. Out of a total of 350.000 STBs, these numbers seems to indicate that a large fraction of STBs are rarely powered off when the customer is not watching TV. Thus, we predict that STB-based statistics may see significant discrepancies between those that only monitor zap events and our new approach.

7. EVALUATION

In this section we describe some of the experiments that we have conducted with AdScorer. We present both a performance evaluation and some interesting observations derived from the AdScorer system. The section is concluded with an evaluation of the techologies used.

7.1 Environment and Experiment Setup

For the experiments, we obtained 1.5 hours of prime time broadcast television sampled from the largest commercial networks, starting at 18:45 on a Thursday evening. Before running the experiments, 23 days of STB data was put through the system up to the point where the experiment started, in order to establish the appropriate statistics.

The experiments involved four servers in addition to the database server keeping track of state. One server was designated event producer, simulating channel zaps obtained from STBs and advertisement identifications obtained from the AdDetector system. Another server were running the message bus, and two more servers were running instances of the *ecaster-core* application, one configured to generate viewer statistics per channel, and the other configured to

score advertisements, as illustrated in the processing section of Figure 1.

Recorded video streams and STB data was used in order to be able to debug and verify system correctness, rather than operating on live video streams and STB data. Moreover, due to time constraints and lack of appropriate video editing tools, the experiments was conducted by simulating the output from the AdDetector system, using manually recorded timestamps and advertisement IDs. However, fingerprints were made from each commercial in one of the commercial breaks of the recordings, using the AdDetector system, and it was verified that the system successfully detected each of them within two seconds when streaming the broadcast recording to system.

Consistency of advertisement scoring results were verified by repeating the tests two times, ensuring that the values were not significantly different from one run to another. While we found slight variations between commercials, the biggest variation we could find was a 0.000349335% disparance between the number of retained viewers for the first advertisement in the first commercial break.

The variations in scores can be attributed to the distributed nature of the system, combined with the event stream processing techniques used, where variations in network and processing latency might lead to slightly different states.

Because channel statistics are generated on a per-second basis on a different machine, small variatons in viewer numbers, as described above is likely to occur. The sequence diagram in Figure 2 should give the reader an intuition for how this can happen, considering that events of type *ChannelStat* and *AdStart* originate from different machines.

The system's ability to handle multiple channels simultaneously was tested by synthetically generating *CommBreak* and *AdStart* events for five different channels at the same time, while the system was receiving live channel zap events. System load was not significantly affected during this experiment. The resulting output files appeared to be correct for the time period sampled, although it was not possible to repeat the experiment with identical values, as the system was operating with live STB data for this particular experiment.

7.2 Scoring Results and Viewer Statistics

Figure 4 illustrates the retained number of viewers for each commercial in a break that started at 18:56:30 on TV2 Norge, divided into regions. The difference in viewer numbers between regions for the most part reflects geographical variations in the number of deployed STBs in the Altibox network. However, there are some relative differences as well, illustrated in Figure 5, where the regional shares of one of the advertisements presented in Figure 4 are displayed.

Although most of the commercial breaks are clearly visible in the viewer plots (Figures 6 and 7), at 18:48, 18:56, 19:23 and 19:54 for TV2, and at 18:53, 19:23 and 19:51 for TVN, the drops in viewer numbers is not nearly as significant as we had expected. Some of this may be attributed to the lack of resolution on STB data, preventing us from accurately capturing channel surfers, as explained in Section 6.1. Moreover, we also note that we can clearly see from Figure 7 that there is a steady growth in viewer numbers over the entire measured interval, except for the significant drop on TV2 at 19:54, and similarly on NRK1 at roughly 20:00.

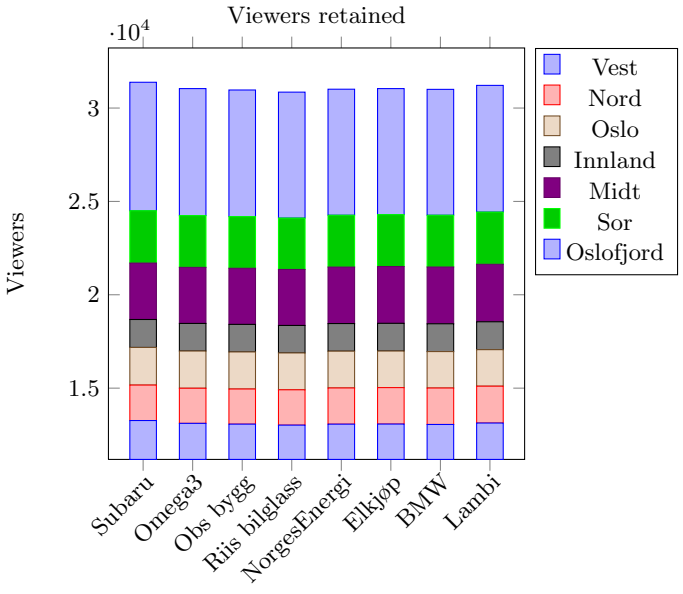

Figure 4: Viewers retained for several commercial spots, split into regions

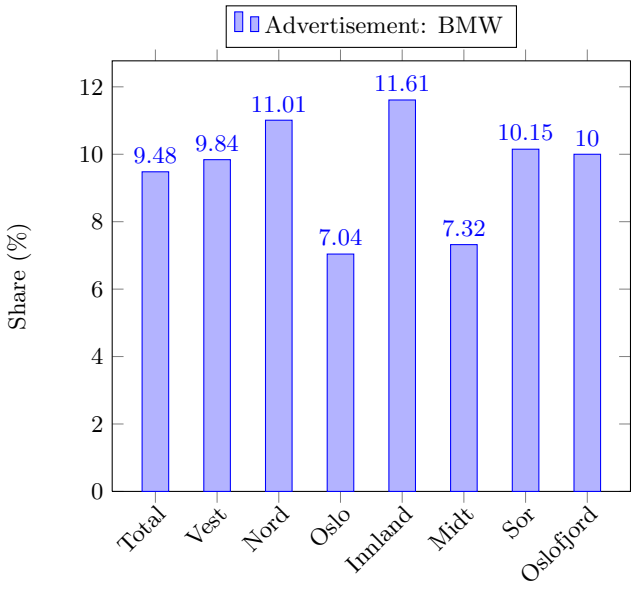

Figure 5: Regional viewer shares for a single advertisement

We note that NRK1 is a non-commercial TV channel, and the largest in Norway.

7.3 Event Middleware Technologies

Here, we give a brief account of our experiences with the various middleware technologies used as building blocks for the event processing and distribution part of AdScorer.

7.3.1 Esper

We found Esper to be very well documented and easy to get started with. One of Esper's most useful abstractions is the *data window*, where one can define a view based on one

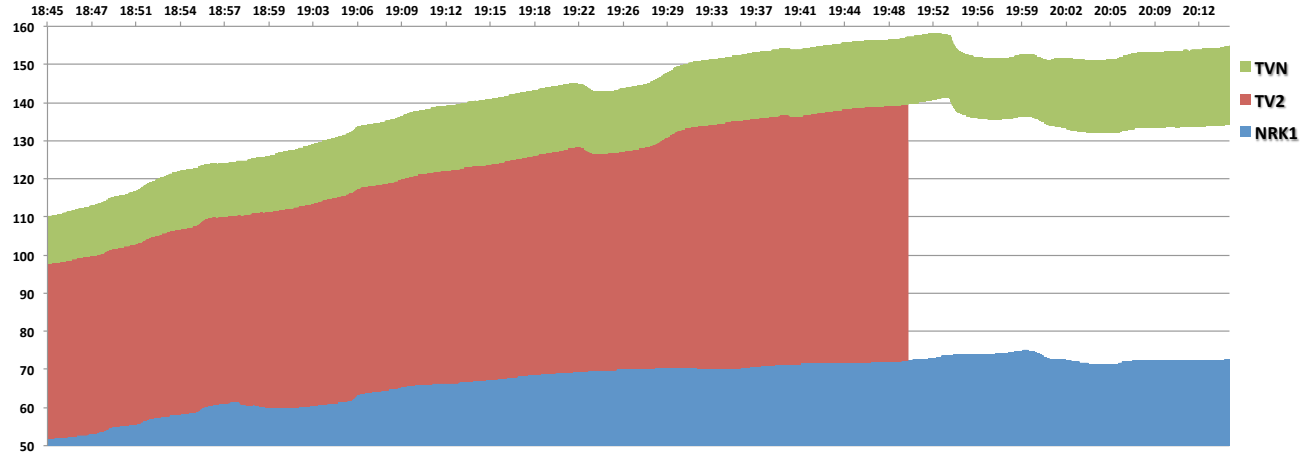

Figure 6: Stacked viewership (in thousands) for the three largest channels, NRK1, TV2, and TVN.

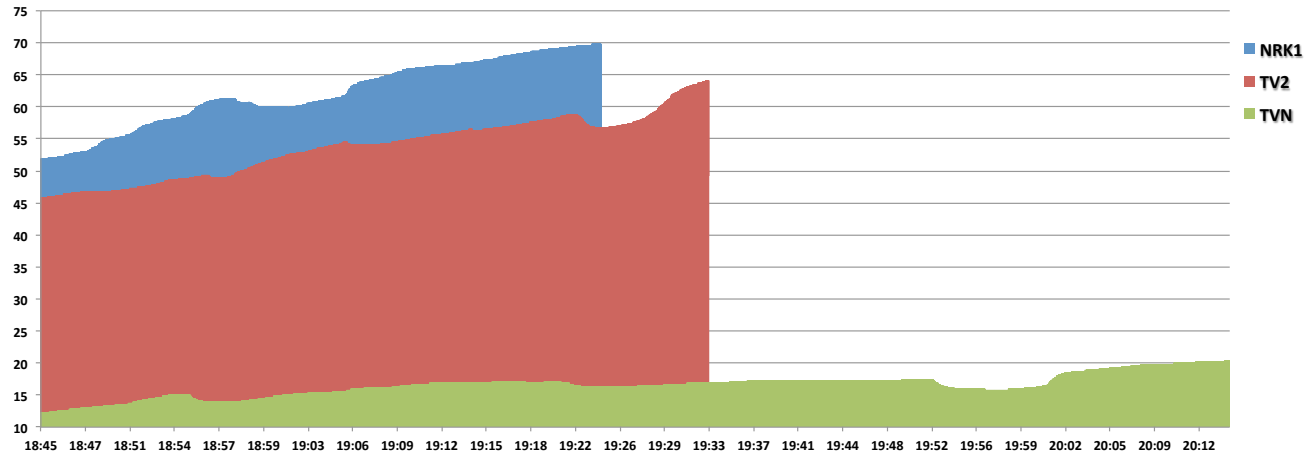

Figure 7: Viewership (in thousands) for the three largest channels, NRK1, TV2, and TVN.

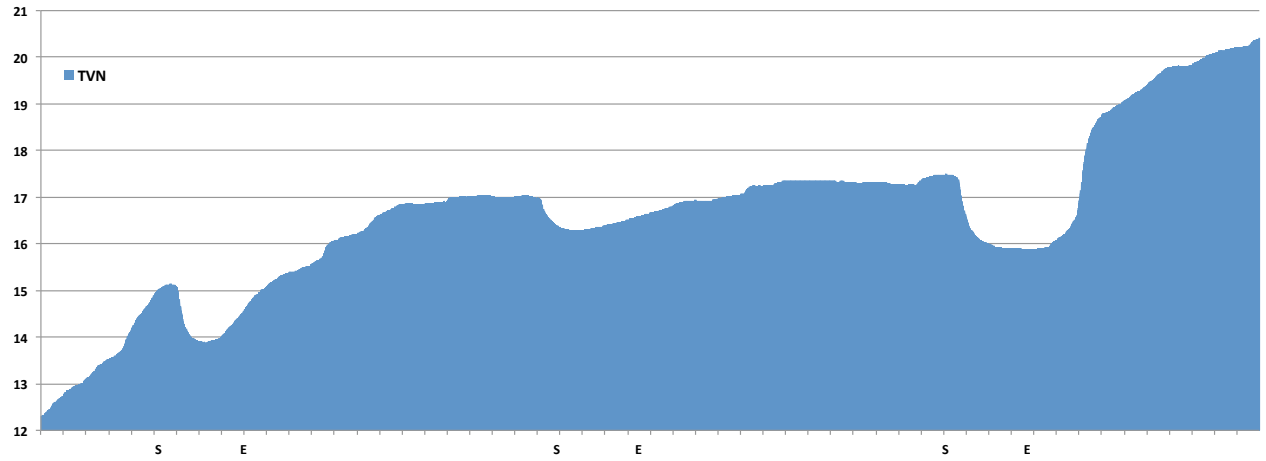

Figure 8: Viewership (in thousands) for TVN annotated with Start and End of commercial break.

Listing 7: EPL to hold last zap event for each STB

```
create window ZapWin.std:unique(ip) as tv.ChannelZap
```

or more properties. For instance, creating a window that holds the last *ChannelZap* event per STB was achieved with a single line of EPL, as shown in Listing 7.

However, we experienced limitations with respect to supported data structures, e.g., to use a *HashSet* to store snapshots of IP addresses tuned to a particular channel at the start and end of an advertisement required a custom aggregation method and an accompanying *Factory* class. Esper currently only supports primitives and the *Map* data type.

7.3.2 HornetQ

Getting started with HornetQ was also fairly simple, and simply a matter of downloading the distribution, making some small changes to the main configuration file and executing the startup script.

Interacting with the HornetQ server was straightforward as well, and required only the inclusion of two JAR libraries on the client side (three, if one needs to use the JMS overlay). Even though the HornetQ distribution includes a JMS overlay, we opted to use HornetQ's native *core* API instead. The core API only offers two abstractions; *queue* and *address*, and the documentation suggests that one can build any interaction included in the JMS specification from these.

There was, however, a learning curve regarding the behavior of the middleware, most notably in the behavior of message acknowledgments and having messages removed from the server after client delivery, and implementing the *publish/subscribe* interaction pattern.

Some of these problems probably could have been avoided by using the JMS overlay, as the overlay seems to take care of many of these behaviors automatically, while the core API to a greater extent leaves it up to the programmer to implement the interactions. The reasons for going with the native API was that it meant one less JAR to depend on and the documentation suggesting slightly better performance and a simpler abstraction. It seems clear now, however, that using the JMS overlay would have been the easier route to take, at least for the *publish/subscribe* interactions.

HornetQ offers excellent performance, and its STOMP interface has proved a convenient way of providing push-style interactions to non-Java client devices.

8. RELATED WORK

Kempe et al [12] argues that audience response should be reflected in the pricing, ordering and selection of ads within a commercial break. In their paper, simple algorithms to measure viewer behavior in response to commercials are presented, and the author's concludes the paper with a more sophisticated algorithm that builds upon the insight gained from these, named the Audience Value Maximization Algorithm.

In their paper on adapting online advertising techniques to television [8], Dorai-Raj and his colleagues at Google also advocates a business model that to a greater extent considers the audience response to advertisements in television, by applying many of the techniques used in online advertising to televised commercials. One of their proposed metrics of viewer response is the IAR algorithm, which are included in the scoring results of AdScorer, presented in Section 5.

An obvious, but important insight presented by Kempe et al [12], is that while online ads can be measured through the positive action of a click, viewers are primarily limited to the negative action of changing the channel in response to televised commercials. The actions of muting/unmuting the audio or turning off the TV or STB are not mentioned by Kempe, or in any of the other papers we reviewed, but also belongs to the current repertoire of viewer responses.

The inclusion of the aforementioned user actions is one of the features of AdScorer that sets it apart from other systems.

At the commercial end of the spectrum, Rentrak [11] appears to be the market leader for STB data aggregation, and is already collecting usage data from millions of STBs deployed by AT&T, Charter, Dish Network and Midcontinent Communications [11]. UK satellite operator BSkyB is another actor in the STB data market that are already collecting STB usage data from over 30 000 devices, correlating these with brand purchasing history from many of the same homes [5].

TRA [4] also combines STB data with credit card transactions, using a third-party blind matching method, where TRA never sees any addresses or names involved in the transactions, in order to measure the effectiveness of advertising campaigns, as well as profiling viewer groups. Data generated from the AdScorer system could also be correlated with purchase activity in the same manner as TRA and BSkyB, and to target ads, like BSkyB intend to do in 2013, provided that privacy laws permits it.

CasterStats [2] provides audience measurement for streaming media, in the form of reports that can be generated through a web interface. However, this appears to be limited to media distributed on the Internet and not broadcast television media, unlike the work presented in this paper.

Coalition for Innovative Media Measurement (CIMM) [3] is a coalition of television content providers, media agencies and advertisers intending on finding new and better ways to measure television media consumption, in the changing media landscape. A main objective of this effort is finding values and applications of STB data, and their contributions include an analysis for the STB data landscape, as well creating and maintaining metrics and an ontology relating to STB data [1].

If CIMM were to succeed in establishing a common standard for STB viewer measurement data, it would greatly benefit all actors who have access to STB data, including Altibox.

9. FUTURE OF MEDIA MEASUREMENT

Because the competition for audience attention has become increasingly intense through the digitization of media, the advertising industry need to continuously improve and re-evaluate its measurement and targeting methods. The reasoning behind this is that information consumes the attention of its audience, and while telecommunication bandwidth is practically infinite, human bandwidth is becoming increasingly scarce [10]. A logical conclusion that can be extracted from this insight, is that television networks should change their business model from selling audience exposure in the form of network time to selling viewer attention, as argued by Kempe et al [12].

Where the traditional mass media channels have established currencies for audience measurement, no such standard currency exists for Internet audiences. An attempt at establishing a standard set of guidelines on how to count impressions was made in 2002 by the Interactive Advertising Bureau, but ended up "extremely confusing and ultimately a compromise" [10]. The main reason for this is the sheer complexity involved in delivering the content.

The difficulties of standardization, however, is unlikely to prevent new models of media measurement from emerging in the near future, as the advertisers and content providers becomes aware of the opportunities of more accurate audience targeting afforded by technologies.

Dorai et al [8] predicts that the online-offline division we have today will soon be replaced by measured-unmeasured as measurement methods converges between different types of media. This view is shared by others as well; Internet and Television is predicted to be measured as one by 2015 in a report [7] published by Forrester Research.

Despite these predictions, the established currencies in mass media audience measurement are unlikely to go away anytime soon, simply because advertisers and media channels needs to agree on a common measure for pricing, even if this is inaccurate [10]. However, the increasing expectations for accountability will create a market for additional, more accurate measurements that can supplement the standard currencies.

The repertoire of viewer actions is likely to grow in the near future, as the media of television gains more interactivity. Examples include interactive links to buy a product, rating possibilities, like/dislike buttons and games.

10. CONCLUSIONS

We have demonstrated a new way of scoring television advertisements that is more is in line with current measurement methods for online media than what is the current practice in the media industry, and more suited for the new models of media consumption.

Our results indicate that our implementation is capable of scoring advertisements on multiple channels simultaneously in near real-time with consistent results, and that event processing is an effective tool for achieving this.

Furthermore, AdScorer is capable of delivering an unprecedented level of detail, not possible through the current measurement regime.

In future work, we aim to complete the implementation of our system, with higher resolution on channel zaps and including volume change, mute and HDMI status events. We are already in the process of developing a graphical front end that operates on live data and displays the scoring results in near real-time.

With this we intend to conduct more detailed viewer behavior analysis in order to derive an improved understanding of the media and to use this understanding to devise new service offerings.

11. ACKNOWLEDGMENTS

We would like to thank Per Fjeld for his valuable contributions in terms of ideas and industry knowledge, and Dagfinn Wåge, Omar Langset and Ronny Lorentzen for facilitating the project.

12. REFERENCES

[1] Cimm lexicon 1.0. Web, May 2010. http://www.cimm-us.org/CIMM_STB_Lexicon_1_May_2010.pdf (accessed 08.11.2011).

[2] CasterStats. Web, 2011. http://www.casterstats.com/(accessed 29.11.2011).

[3] Coalition for innovative media measurement. Web, November 2011. http://www.cimm-us.org/about.htm.

[4] TRA Global. Web, 2011. http://www.traglobal.com/(accessed 27.11.2011).

[5] Bskyb preparing for linear targeting, scheduled for spring 2013. Website, 2011. http://www.v-net.tv/bskyb-preparing-for-linear-targeting-in-spring-2013/ (accessed 30.05.2012).

[6] Meeyoung Cha, Pablo Rodriguez, Jon Crowcroft, Sue Moon, and Xavier Amatriain. Watching Television over an IP Network. In *IMC*, 2008.

[7] David M. Cooperstein, Kim Le Quoc, and Jean-Yves Lugo. The future of media measurement. Web, January 2010.

[8] S. Dorai-Raj, Y. Interian, I. Navernioukk, and D. Zigmond. Adapting online advertising techniques to television. *Online Multimedia Advertising: Techniques and Technologies*, page 148, 2010.

[9] Pål Evensen and Hein Meling. A paradigm comparison for collecting tv channel statistics from high-volume channel zap events. In *DEBS*, pages 317–326, 2011.

[10] Marissa Gluck and Meritxell Roca Sales. The future of television? advertising, technology and the pursuit of audiences. Web, The Norman Lear Center, University of Southern California, September 2008. http://www.learcenter.org/pdf/FutureofTV.pdf.

[11] David Goetz. Mpg signs rentrak deal for set-top-box data to help with upfront planning. Web, April 2011.

[12] D. Kempe and K.C. Wilbur. What can television networks learn from search engines? how to select, price, and order ads to maximize advertiser welfare. Technical report, Working paper, Viterbi School of Engineering, University of Southern California. http://ssrn.com/abstract1/41423702, 2009.

[13] Karl Philip Lund. Gamle målemetoder! Web, October 2010. http://www.kampanje.com/kommentert/article5772345.ece (accessed 25.11.2011).

[14] Apache maven. Website, 2012. http://maven.apache.org/ (accessed 22.05.2012).

[15] Magna: Tv ad spending on the upswing for foreseeable future. Website, 2010. http://www.mediapost.com/publications/article/129755/ (accessed 30.05.2012).

[16] Nielsen Ratings. Web, 2011. http://en.wikipedia.org/wiki/Nielsen_ratings.

[17] Opher Etzion and Peter Niblett. *Event Processing In Action*. Manning, August 2010.

[18] Statistisk sentralbyrå (statistics norway). Website, 2012. http://www.ssb.no/familie/ (accessed 26.05.2012).

[19] TNS Global Market Research. Web, 2011. http://www.tnsglobal.com/.