Repurposing Web Analytics to Support the IoT

Mateusz Mikusz, Sarah Clinch, Rachel Jones, Mike Harding, Chris Winstanley and Nigel Davies
School of Computing & Communications
Lancaster University, Lancaster, UK
m.mikusz | s.clinch | r.jones1 | m.harding | c.winstanley | n.davies @ lancaster.ac.uk

ABSTRACT
The widespread use of free analytics tools has helped revolutionise the web – enabling developers to gain deep insights into user behaviour. Analytics are also perceived as critical to enabling the next generation of the Internet of Things. However, despite the existence of numerous IoT analytics engines none have had the catalytic effect of web analytics in helping to transform developers' understanding of the systems they create. In this paper we report on our experiences of creating and using a system that looks to repurpose web analytics to enable growth in the future IoT.

INTRODUCTION
Web analytics have helped revolutionise the development and use of web-based applications and services. Services such as Google Analytics are freely available and provide developers and website owners with comprehensive data on how users interact with their sites. In particular, web analytics go far beyond simple monitoring of attributes such as "hits" on a website and allow detailed analysis of user behaviour – enabling sites to be rapidly adapted to address interaction problems, thus helping to maximise the potential for users to achieve the desired goal.

Drawing significantly on web technologies, the Internet of Things (IoT) is emerging as an increasingly important paradigm for creating a world of connected devices. Compelling use cases have been articulated in domains including health, transport, logistics, domestic energy control and "smart' cities. These use cases have been supported by the development of a wide range of technologies in each domain [1]. To provide core support for the IoT a significant number of distributed platforms have been developed in both academia and industry. These platforms offer a range of services from simple IoT device brokering to complex data cleansing and prediction capabilities. However, while significant work has been carried out on the plumbing necessary for the creation of the IoT, significantly less research has been invested in the creation of tools for understanding the data created by the IoT. Data visualisations and analytics often tend to be optimised for a single domain (e.g. smart cities [3]) or form part of complex IoT infrastructures that are in marked contrast to the ease of use and ubiquity of web analytics.

In this paper we explore the concept of repurposing an existing, widely available, web analytics system for use in the IoT domain. We describe a simple cloud-based mapping service that can translate IoT events into corresponding Google Analytics events and we provide detailed examples that illustrate the use of such a service. In particular, we show how generic web analytics can be used to support a number of distinct IoT analytics use cases ranging from simple device and sensor data monitoring to complex user interactions with multiple IoT objects.

While we do not believe that web analytics provide the total solution for IoT data visualisation and reporting, in this paper we show that current platforms can be successfully repurposed in a range of scenarios, enabling the emerging IoT to leverage the significant investment in today's global web analytics services.

THE ANALYTICS LANDSCAPE

Web Analytics
Historically web analytics were dominated by server-side analytics based on access log files. Modern web analytics include comprehensive client-side data collected using JavaScript embedded in each web page. Each type of user interaction can be described using specific attributes such as "page views" and "events" consisting of different features (table 1). To enable general tracking of browsing activity, web analytics uses page views—each time a user opens a page, a page view event is reported to the analytics service. Page view events consist of information about the page visited (title, host and full path), the path of the referring website, and can be optionally extended by a unique user identifier for cross-device tracking. If no user identifier is specified, the backend recognises users through client-side cookies allowing the analytics service to track the user across multiple pages and provide visualisations of navigation patterns. Additionally other information such as the number of page views, unique and returning visitor numbers and referral paths is visualised.

Customisable "event" hit types can be used to express on-page events. Each event hit type consists of four attributes: category (required), action (required), label (optional) and value (optional). Together the category and action attributes describe the action that was performed on the page—for example, a user clicking on a play-button on the page. If a value is supplied, it must be a non-negative integer; the analytics backend dynamically generates appropriate graph visualisations for incoming values and displays the number of incoming events in an overview. Users can filter for specific categories, actions and labels and specify goals to be met based on the value attribute.

Client-side analytics data is typically communicated to a cloud-based analytics service using a protocol such as the Universal Measurement Protocol (UMP) [6]. In addition to events and page views, UMP supports an additional set of hit types for mobile applications, e-commerce systems, and social interactions, and can be extended using custom dimensions and metrics.

IoT Analytics

Much of the value of the Internet of Things is expected to be derived from the capture and analysis of data from embedded sensors. In the commercial world IoT applications often represent complex, end-to-end vertical systems that integrate both hardware and software components in domains such as transport, industrial automation and smart cities [1, 7].

Enterprise cloud-based IoT platforms such as Mnubo (`http://mnubo.com/`), SAP IoT Solutions (`http://go.sap.com/solution/internet-of-things.html`) and Thingworx (`http://www.thingworx.com/`) aim to ease development and deployment by reducing the complexity, time and cost required to implement IoT applications, providing off-the-shelf services including data storage, machine learning and data analytics. However, in contrast to web analytics, analytics for the IoT is a heavily fragmented market place with a large number of players offering a diverse set of capabilities. Several state-of-the-art IoT cloud platforms such as AGT (`https://www.agtinternational.com/iot-analytics/iot-analytics/iota-agts-iot-analytics-platform/`) provide integrated IoT analytics capabilities but these are typically closed systems. Platforms such as ParStream (`https://www.parstream.com/product/`) provide analytics for IoT applications but require developers to integrate with proprietary data storage to generate analytics over incoming data streams. Similarly, Intel's IoT Cloud Analytics (`https://software.intel.com/en-us/iot`) platform is compatible with Intel accredited IoT devices (i.e. sensors) and agents.

In addition to general IoT analytics platforms, researchers have also developed analytics tools for specific domains. For example, for signage analytics and audience measurement Intel's AIM suite (`https://aimsuite.intel.com/`) provides a visual analytics tool for realtime audience tracking capturing both demographics and behaviour. A similar system by IBM enables the creation of behaviour analysis reports for people inside buildings or public spaces [8]. In domains such as smart cities researchers have created visualisations that enable the public to understand the flow of information within their city [3].

Analysis

There are marked differences between the web and IoT analytics landscapes. In the former there is a dominant player with a freely available offering used by over 60% of the Fortune 500 websites regardless of their application domain (e.g. e-commerce, blogs, news sites). The development overhead for incorporating analytics into a website is very low and most developers are familiar with the steps required. Moreover, understanding the basic analytics provided is easy and there exists a large community of specialists able to provide more detailed analysis. In contrast, the IoT domain is characterised by complex platform and analytics offerings and as a result many developers end up developing their own analytics dashboards using visualisation toolkits such as D3.js (`d3js.org`). The emergence of research into pervasive analytics (e.g. [2]) and analytics for the IoT [5] demonstrates new application domains for data analytics but has not explored the repurposing of an existing web analytics system to provide the necessary capabilities.

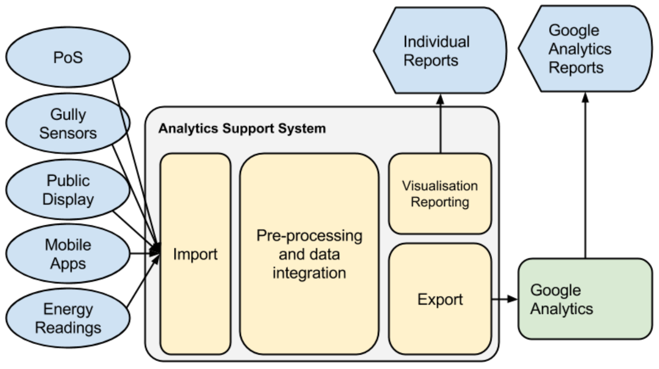


Figure 1: Pheme architecture diagram.

In this paper we consider whether existing web analytics services can be repurposed to support a reasonable subset of IoT application domains and associated analytics requirements. If such repurposing can be shown to be viable then IoT developers could benefit from an existing analytics platform that can clearly operate at global scale, has a low barrier to entry, and brings a large user base that understands how to use the tools and interpret the results. Perhaps more critically, such an approach would lay the foundation for the development of tools to explore user interaction with the IoT. As such our work is designed to help accelerate the adoption of the IoT through the use of web technologies and builds on a significant body of research on systems such as Cooltown [10] that explored the integration of web technologies with people, places and things in the physical world and that continues to have significant traction in the community [11].

PHEME: AN ANALYTICS REPURPOSING SERVICE

Overview

We have designed and developed a cloud-based service, Pheme, that allows existing web analytics systems to be repurposed for data collection in the IoT. Our architecture consists of four components (Figure 1): an *Import* module, a *pre-processing and data integration* module, a *visualisation and reporting* engine, and an *Export* module.

Page Views	Events
Time on site	Category, action, label, value
Bounce rate	Grouping across attributes
Funnels	Non-negative integer values
Entry & exit points	Graphs visualisation for values
Content	Link events to pages
Heirarchical URI / drill down	
Value is 1	
Referrer	
Graph visualisation for page views	
User ID, Time	

Table 1: Comparison of features provided by Google Analytics when using Page View and Event hit types.

The flow of data through Pheme is as follows. Client devices and sensors report their data to the *Import* module using an extension to UMP that provides a richer set of data types for describing interaction with IoT components. In common with most web analytics systems, in order to correctly associate incoming data with user accounts every UMP message must include a tracking ID obtained from Pheme through a simple user interface.

Each incoming dataset triggers *pre-processing and data integration* modules that parse incoming data, creating objects for use by other components of the system (e.g. for data validation or filtering). A modular design allows Pheme to be extended with additional pre-processors at any time; for example, modules may be written to fuse datasets from multiple data sources, enable cross-device analytics or enrich datasets based on historical data.

Following pre-processing and integration, data may then be automatically exported and injected into third-party analytics engines (e.g. Google Analytics). The *Export* component allows developers to plug in multiple "injectors"—each of which provides a specific mapping to make the data compatible with a third-party service. Since each user may have multiple tracking instances with unique tracking IDs registered to Pheme, each instance can be associated with one or more injectors. For example, multiple injectors representing different mappings can be used to provide support for different reporting requirements simultaneously.

Finally, if the third-party web analytics service does not provide sufficient visualisations and reporting, the data may be fed to Pheme's *visualisation and reporting* engine to provide specific extensions to visualise the pre-processed data. In this way we utilise existing web analytics reporting tools but can also support custom IoT-specific reporting as needed.

Implementation

Pheme is implemented in Python on the Google AppEngine cloud service. The import module provides a RESTful API that accepts extended UMP data. Different event and reporting types are distinguished using specific UMP attributes—all events are reported to a single API.

The pre-processing and export of data is implemented through provision of a set of injectors. Registering new mappings with the system consists of creating a new Python class in the backend by inheriting from provided base classes and simply writing the mapping. For the end user, we offer a user interface to set up their own tracking IDs and to configure each ID with appropriate injectors. During this assignment users can provide a third-party tracking ID (e.g. for Google Web Analytics) to be used for every dataset injected to the service. Two injectors are currently supported. The "Datastore Injector" writes each dataset to a local datastore within Pheme. The "Google Analytics Injector" maps datasets to web analytics terminology and pushes these in real-time to Google Analytics using vanilla UMP. Injector assignments can be removed by the user. For example, removing the Datastore Injector will turn Pheme into a simple analytics mapping proxy and prevent it from storing the raw data.

To allow easy integration into existing IoT deployments, we provide client libraries for Python 2.7 and 3, PHP, and JavaScript. These libraries provide a common set of event types and methods for reporting to Pheme.

As an example, for analytics tracking in Python, the library is first initialised by providing a unique tracking ID (previously registered with the system using the backend UI): `analytics = IoTAnalytics('tracking id')`. The library then generates a client UUID to be included with each request; optionally, this ID may be overwritten by the user to, for example, track corresponding values across multiple devices and sensors.

In order to submit relevant data, for example, to track specific content, the "track page view" method can be used to provide parameters about the document: `track_pageview(location, host, path, title, description, id, hash)`. Alternatively, users may use the event type to track sensor values: `track_event(category, action, value, label)`. Choosing the correct method and attributes is important to achieving the right mappings and reporting (described in more detail later).

CASE STUDIES OF USE

We have used our generalised mapping service to support analytics in four distinct IoT application domains.

IoT Energy Sensors

Our first example focuses on the use of web analytics to provide visualisation of IoT energy sensors used in domestic energy consumption research. While smart meters can provide an overall measure of the power consumption of a household, fine-grained insight into energy use requires the deployment of multiple sensors – ideally one for each plug in a household. The researchers we collaborated with wished to have an analytics system that supported a range of sensor types, named groups of sensors, and fault monitoring. The Plugwise domestic energy sensors used transmit readings wirelessly to a gateway device in each household; this gateway was modified to include calls to Pheme's client library that reported each sensor reading.

Each sensor reading consists of a MAC address, user-specified sensor name, a timestamp, and the current power consumption (a float value). For analytics provision, these sensor readings must be mapped onto appropriate Google Analytics parameters. One possibility would be for each sensor reading to be mapped onto one "page view". This would allow tracking of whether sensors are "alive" and reporting values, but the values themselves would not be expressed through this type. Instead we use the "event" type with its user-defined attributes: *category*, *action*, *label* and *value*.

We map each sensor's MAC address to the *client ID* attribute. In this way the number of sensors reporting to the analytics backend maps directly to the number of "active users on site". Each user-specified sensor name is mapped to the *label* attribute such that the Google Analytics backend represents each sensor appears under a friendly name for easy navigation. To report sensor readings themselves we map reported power consumption onto the *value* attribute. However, since Google Analytics only allows positive integer values for this attribute, Pheme scales and rounds float values. Sensors are grouped using *category* value; for example, categories such as "bedroom" and "kitchen" can allow the user to view the cumulative power consumption in these rooms.

This mapping provides simple yet useful analytics features for researchers with support for real-time and historical viewing of sensor data. Users can easily determine whether all sensors are reporting through the total number of "active users on site". Each of the event values can be directly viewed to, for example, determine the cumulated energy consumption for a given time frame. Erroneous sensor readings can be automatically detected in the analytics service by specifying and checking progress against goals for incoming event values.

IoT Cloud Data Services

A common IoT use case involves adding value to sensor data through additional (cloud-based) processing; for example, data cleansing or combination of multiple data sources and historic traces in order to predict future sensor states. In this example we focus on the use of analytics to provide insights into a system designed to report current and predicted future states of IoT sensors.

Our use case focuses on environmental monitoring, in particular the monitoring and prediction of river water levels. We wanted to provide an analytics service that provided a view on the data being produced by river level sensors and the activities of a custom IoT hub that provided a prediction capability for these sensors. In particular, we needed to support threshold monitoring of sensor values, spatial and temporal reporting and fault monitoring.

Where the previous use case focussed on raw sensor data as input to the analytics service, here the problem centres on understanding processed IoT data using analytics. For this reason, the analytics library was integrated directly into the prediction hub. Rather than have each each sensor separately report raw data, each time a new prediction is generated, the hub reports to Pheme with the predicted sensor state.

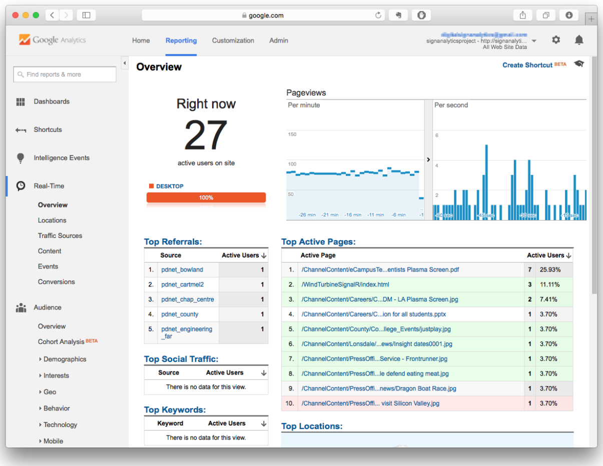
We again reported sensor readings and predictions as *events* rather than page views. In contrast to the previous example we chose to map river names to the *category* attribute, sensor identifiers to the *label* attribute, and predictions as *action*s. This enabled us to use the *client ID* to report the hub producing the prediction, and to allow filtering of predictions for specific rivers and sensors. This mapping gives an insight into the overall number of predictions (expressed as "active users on site") created in a specified time frame. River levels themselves are mapped onto the event's *value* attribute—enabling Google Analytics to compute an average values and graph visualisations; scaling was again used to overcome the limitation of only being able to report integer values.

A common feature of web analytics is the geographic mapping of requests based on IP. At present, manipulation of the location attribute for requests is not supported through UMP. Proxying can allow the IP address to be set in order to generate a location, but the accuracy of the reverse IP lookup is likely to be poor and in many cases a sensor's location is known or measured by the sensor itself. In order to address this, our extended UMP includes attributes for latitude and longitude. A custom visualisation on our analytics backend maps out the sensor data based on these additional attributes.

Sign Analytics

While our first two use cases focused on traditional IoT analytics we now explore the use of Pheme to support reporting of both IoT analytics and user engagement. Our example is based on the increasingly pervasive digital signs that illustrate the way in which the presence of IoT devices can change how people engage with space. Analytics offer value to sign owners by providing insights into the operation and success of a deployment (e.g. content shown, level of audience engagement).

We have operated a campus-wide signage network for over ten years and in this use case we describe how we have exploited web analytics to provide simple signage analytics to report faults, content impressions, and user interactions. By integrating Pheme's client library calls into existing open



Top Referrals:

	Source	Active Users ↓
1.	pdnet_bowland	1
2.	pdnet_cartmel2	1
3.	pdnet_chap_centre	1
4.	pdnet_county	1
5.	pdnet_engineering_far	1

Top Social Traffic:

Source	Active Users ↓
There is no data for this view.	

Top Keywords:

Keyword	Active Users ↓
There is no data for this view.	

Top Active Pages:

	Active Page	Active Users ↓	
1.	/ChannelContent/eCampusTe...entists Plasma Screen.pdf	7	25.93%
2.	/WindTurbineSignalR/index.html	3	11.11%
3.	/ChannelContent/Careers/C...DM - LA Plasma Screen.jpg	2	7.41%
4.	/ChannelContent/Careers/C...ion for all students.pptx	1	3.70%
5.	/ChannelContent/County/Co...llege_Events/justplay.jpg	1	3.70%
6.	/ChannelContent/Lonsdale/...ews/Insight dates0001.jpg	1	3.70%
7.	/ChannelContent/PressOffi...Service - Frontrunner.jpg	1	3.70%
8.	/ChannelContent/PressOffi...le defend eating meat.jpg	1	3.70%
9.	/ChannelContent/PressOffi...news/Dragon Boat Race.jpg	1	3.70%
10.	/ChannelContent/PressOffi... visit Silicon Valley.jpg	1	3.70%

Figure 2: Using Google Analytics for visualising currently played content items on digital signs.

source signage software, we developed a sign analytics system to monitor a campus deployment of approximately 30 displays [4].

To describe content shown on a display, we report each transition of a content item onto the display as a *page view* for that item. This enables Google Analytics to report the typical content duration for media (across a deployment or single sign), as shown in Figure 2. In addition, *user funnels* can be used to visualise content change patterns (for example, identifying when display owners typically create a schedule in which a specific content item is always followed by another specific item).

Video analytics are a useful tool to understanding the real impact of displays (for example, to count the number of people looking at a display) and can be implemented using relatively simple image processing software (e.g. OpenCV). We identified two approaches to handling this data. In the first, each passerby looking at a display is classed as one "page view". Arguably the most natural mapping, this means that reports of "active users" directly correlate to people currently observing the screen. However, this approach makes it impossible to simultaneously report content changes as page views. The second approach instead reports face counts as custom events, thereby maintaining compatibility with the previously described technique of reporting content using page views. Our implementation uses this second approach, which also offers the potential for additional reporting (e.g. of demographics, gesture patterns) through the addition of new event categories.

To understand the pattern of content shown on a display, and to provide support for monitoring and reporting, it is necessary to make displays visible as first class entities in the analytics reporting tools. However, in most web analytics platforms users generating the hit data are typically anonymised and invisible, meaning that the origin of a request (i.e. the display) would not be visible in our display analytics. To overcome this, we introduce an additional mapping from display name to the *referrer* attribute. Each display is additionally mapped as a *user* and *client ID* in the analytics backend meaning that the "total number of (active) users" correlates to the count of distinct displays, and the number of *unique visitors* to the number of (active) displays.

For additional reporting we use custom events for information such as the physical power state of the signs. Combining this physical power state information with page views within Pheme can lead to a more reliable page count (e.g. "only report page views if the display is turned on").

Finally, we note that the use of "goals" can provide automated monitoring. For example, setting goals for page view counts can automatically identify whether content item has been shown the expected number of times.

Multi-Device Interaction

As the IoT becomes more pervasive, user interaction with the IoT will span not one but many devices. Analytics provide an opportunity to understand the relationships between interactions with multiple distinct IoT devices. Our final use case focuses on how users engage with multi-screen ecosystems, and how attention flows transfer between devices during those engagements.

To explore multi-device engagement monitoring we created a system called ENGAGE [9] that provides a set of independent *engagement sensors*. We equipped laptops and digital signs with web cameras and OpenCV face counters to determine the level of user engagement. For engagement monitoring of smartphones we developed a simple simple background application. We integrated these ENGAGE tools with Pheme for basic reporting capabilities. Each device reported the face count, interaction or content change using an extended UMP hit type "device interaction" consisting of the application or content name, path, device type, user ID, type of interaction ("touch" or "move") and the physical state of the device (whether the screen was turned on).

We generated a mapping for the translation of multi-device engagement flow events to traditional web-analytics terminology. First, to support cross-device tracking of the same individual, a unique user ID was shared accross devices and mapped onto both the *user ID* and *referrer* attribute of Google Analytics. Thus, current engagements correlate directly to one session and can be retrieved as "session time." Second, we mapped engagement and interaction with devices and applications as *page views*. This enables us to take advantage of the "user funnel" created automatically by Google Analytics for visualising cross-device interaction patterns.

Within the page view model, we composed the content URI using a hierarchical scheme (`device_type/application_name/content` thus allowing "content drilldown": users could first look at all device types in general, having the option to drill down into more detail. New device types can be dynamically created and reported to the analytics backend without any further mapping rules. Although not currently pushed to Google

Analytics, detailed information about interaction type is also collected by the client library.

Scalability

One advantage of leveraging web analytics is the ability to leverage a highly scalable infrastructure. Google Analytics itself has a large user base and is capable of handling 1 billion hits per month per property (limited to 10 million for non-premium accounts) and 200,000 hits per user (e.g. client or sensor) per day (**https://developers.google.com/analytics/devguides/collection/gajs/limits-quotas**). The latter equals to approximately 2.31 hits per user per second. Deployed as a cloud service, Pheme was built with scalability in mind and benefits from the scalability of App Engine including unlimited storage and the ability to handle large numbers of requests. To date, Pheme has run for eight months supporting our signage use case with an average of 30 clients generating ~114,000 requests per day (~1.3 requests per second, and a total of 97 GB of data). Our environmental monitoring use case featured over 3,100 water level sensors.

CONSIDERATIONS WHEN USING WEB ANALYTICS FOR IOT DATA

Our experiences with four distinct use cases suggest that when choosing to use a web analytics service for IoT applications a number of detailed considerations must be borne in mind. A key design decision is the selection of either *page view* or *event* hit types to represent data (the features of these hit types are described in Table 1). If the data to be recorded is not binary then the lack of a value attribute for page views dictates that the event type should be used. Multiple variable values for the same entity can be mapped using separate events, each event sharing a common label (i.e. sensor identifier) and having a unique action that describes the value type. The values themselves can be mapped onto the value attribute (but must be given as positive integers). Thus, additional readings and states can be added in the future by pushing a new action/value pair with the same label. An event's category attribute can be used for grouping multiple entities (e.g. all sensors in a household).

By contrast, if the IoT data simply reflects an occurrence of an event (where the important points are the start and end of that occurrence) then page views may be more appropriate; using page views in this way gives access to additional aggregations (e.g. time on site, bounce rate, funnels). For example, using page views in a domestic energy setting might allow observation of patterns in device use, such as the turning on of a kettle frequently being followed by use of the oven. However, the submission of a new page view event has the effect of ending any previous page view from the same user; in some cases household energy events may occur simultaneously (i.e. when the kettle and toaster are running concurrently) and this is not well represented by page views.

Page views may also be an appropriate reporting mechanism if the IoT entity being monitored can be described as a hierarchical association of elements that can be mapped onto a url-style path. For example, sensors installed in a certain room of an apartment could be described as `apartment/room/sensor` enabling content drill-down through these different levels.

Care should be taken when considering using web analytics to monitor IoT event data that is not reported in a timely fashion. Specifically, Google Analytics allows small time offsets to be reported using UMP to enable capture of events during periods of temporary disconnection (up to 4 hours). However, Google Analytics does not allow historical data or future, predicted events to be reported. Our extensions to UMP in Pheme do allow such events to be reported for local visualisation but these events cannot easily be injected into Google Analytics.

Web analytics aggregate and anonymise individual user (client) behaviours. However, when used in the IoT domain, it is common to want to filter and report hit events based on a specific client ID. Since this is unsupported, we found a common practice across our use cases was the mapping of sensor/device ID to both client ID *and* referrer – thus making the entity's underlying identifier visible in the web analytics UI.

We note that the provision of client libraries allowed easy integration of analytics into IoT-related services similar to the JavaScript snippet approach for web analytics. A standard set of methods allows developers to track page views or events; mapping is performed both client-side (by deciding which values should be reported as events or page view paths) and on Pheme (by mapping onto standard UMP and injecting data into Google Analytics).

CONCLUSIONS

Web analytics have revolutionised the development and use of web-based applications and services. In this paper we have shown that these same technologies can be repurposed to support IoT applications. In particular, through four distinct use cases we have demonstrated the implementation of a range of analytics services including sensor monitoring and user engagement tracking that have been successfully used over a number of months in an IoT deployment. Clearly Pheme does not provide the type of detailed domain-specific analytics that a comprehensive IoT analytics package offers. Moreover, our decision to use a RESTful API for Pheme coupled with the inherent limitations of the analytics system that we use means that we have limited support for streaming data and that data may require processing prior to injection (e.g. to accommodate negative numbers). However, we are able to add useful analytics to IoT deployments with minimal development effort and cost – thus addressing a significant barrier to widespread implementation of the IoT. Where more sophisticated analytics are required the architecture of Pheme enables data to be easily routed to domain specific visualisation and reporting tools.

In addition to providing a generally useful approach to IoT analytics, we believe that considering IoT in the context of web analytics will also help trigger a debate on the form that user-oriented analytics will take in the future IoT. If we are able to successfully capture user interactions with the IoT at scale this will enable developers to rapidly refine their device and application designs to meet user needs – helping to enable the future IoT.

ACKNOWLEDGMENTS

This research was partially funded through FET grant number: 612933 (RECALL) and was made possible with the support of a Google Faculty Research Award.

REFERENCES

1. Atzori, L., Iera, A., and Morabito, G. The internet of things: A survey. *Comput. Netw. 54*, 15 (Oct. 2010), 2787–2805.
2. Balan, R. K., Misra, A., and Lee, Y. Livelabs: Building an in-situ real-time mobile experimentation testbed. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, ACM (2014).
3. Batty, M., and Hudson-Smith, A. Visual analytics for urban design. *Urban Design 132* (2014).
4. Friday, A., Davies, N., and Efstratiou, C. Reflections on long-term experiments with public displays. *Computer, IEEE 45*, 5 (May 2012), 34–41.
5. Funk, M., van der Putten, P., and Corporaal, H. Analytics for the internet of things. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, ACM (2009), 4195–4200.
6. Google Inc. Measurement protocol parameter reference. `https://developers.google.com/analytics/devguides/collection/protocol/v1/parameters`, March 2015.
7. Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. Internet of things (iot): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst. 29*, 7 (Sept. 2013), 1645–1660.
8. Hampapur, A., Bobbitt, R., Brown, L., Desimone, M., Feris, R., Kjeldsen, R., Lu, M., Mercier, C., Milite, C., Russo, S., Shu, C.-F., and Zhai, Y. Video analytics in urban environments. In *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance* (2009), 128–133.
9. Jones, R., Clinch, S., Alexander, J., Davies, N., and Mikusz, M. Engage: Early insights in measuring multi-device engagements. In *Proceedings of the 2015 International Symposium on Pervasive Displays*, ACM (2015).
10. Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., and Spasojevic, M. People, places, things: Web presence for the real world. In *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications*, IEEE Computer Society (2000).
11. Want, R., Schilit, B., and Jenson, S. Enabling the internet of things. *Computer 48*, 1 (Jan 2015), 28–35.