SPF: An SDN-based Middleware Solution to Mitigate the IoT Information Explosion

Mauro Tortonesi[1], James Michaelis[2], Alessandro Morelli[1], Niranjan Suri[2,3], Michael A. Baker[2]
[1]Department of Engineering, University of Ferrara, Ferrara, Italy
mauro.tortonesi@unife.it, alessandro.morelli@unife.it
[2]United States Army Research Lab, Adelphi, MD, USA
james.r.michaelis2.civ@mail.mil, niranjan.suri.ctr@mail.mil, michael.a.baker.mil@mail.mil
[3]Florida Institute for Human and Machine Cognition, Pensacola, FL, USA
nsuri@ihmc.us

***Abstract*—Managing the extremely large volume of information generated by Internet-of-Things (IoT) devices, estimated to be in excess of 400 ZB per year by 2018, is going to be an increasingly relevant issue. Most of the approaches to IoT information management proposed so far, based on the collection of IoT-generated raw data for storage and processing in the Cloud, place a significant burden on both communications and computational resources, and introduce significant latency. IoT applications would instead benefit from new paradigms to enable definition and deployment of dynamic IoT services and facilitate their use of computational resources at the edge of the network for data analysis purposes, and from smart dissemination solutions to deliver the processed information to consumers. This paper presents *SPF* (as in "Sieve, Process, and Forward"), an SDN solution which extends the reference ONF architecture replacing the Data Plane with an Information Processing and Dissemination Plane. By leveraging programmable information processors deployed at the Internet/IoT edge and disruption tolerant information dissemination solutions, SPF allows to define and manage IoT applications and services and represents a promising architecture for future urban computing applications.**

Keywords—Internet of Things; Software Defined Networking; Information Dissemination; Value of Information.

I. INTRODUCTION

A recent white paper from Cisco Systems Inc. predicts that by 2018 Internet-of-Things (IoT) devices will generate more than 400 ZB of data per year [1]. Consequently, a multitude of IoT applications are expected to be deployed, each designed to process data from IoT and mobile devices in urban environments. However, IoT applications, especially those designed to operate in complex urban computing environments [2], require sophisticated computational and networking architectures, so their realization presents significant challenges.

From a system design perspective, handling the large volume of information generated by IoT devices is going to be an increasingly relevant issue. In fact, next-generation applications will primarily run on mobile devices that operate in a challenging, heterogeneous, and dynamic wireless networking environment. They will, however, be required to provide enough storage and computational power for the processing of information as well as to create and manage communication paths from information producers to information processors and finally to information consumers, while at the same time orchestrating the various application entities involved.

However, many approaches realized so far, based on the collection of IoT-generated data for storage and processing in the Cloud, present several drawbacks. First, the cost of storage can be very high and the latency in information delivery can be substantial [3]. Second, the load on the network can be excessively high. Finally, these approaches mostly consider large one-time processing of large volumes of data, which is not scalable. Rather than attempt to store and process *all data* available in an IoT ecosystem, more practical approaches will likely rely on selective filtering to aid in data storage and dissemination.

The dissemination of processed information also presents some challenging issues. In the urban computing environment, characterized by heterogeneous and dynamic wireless networking, the stakeholders using and managing IoT applications cannot assume to be in control of the networking infrastructure or even leverage TCP flows for efficient information dissemination. Instead, applications need specific middleware support to automatically take advantage of communication optimization opportunities such as those offered by Device-to-Device (D2D) communications and mobile offloading techniques (i.e., switching from 4G/LTE to WiFi communications in case an open hotspot is detected) [6].

We believe that these challenges could be effectively addressed by adopting an SDN approach that considers information processing and dissemination functions at the same time, which represent two fundamentally interrelated aspects of IoT applications. This could be achieved by extending the reference SDN architecture proposed by the Open Networking Foundation, replacing the Data Plane with an Information Processing and Dissemination Plane.

This paper presents *SPF* (Sieve, Process, and Forward), an SDN solution that aims to address the explosion of IoT data by processing it at the edge of the network, in close proximity to the source of its generation. In order to filter information objects, SPF uses a minimum content difference threshold for

new IoT data to be considered for processing and dissemination. In addition, SPF prioritizes dissemination of critical information by ranking objects according to their corresponding Value of Information (VoI) metric [5].

SPF is a middleware level solution that does not require any control of the network infrastructure (e.g., 4G/LTE networks) in order to be deployed. SFP relies on robust and disruption-tolerant information dissemination solutions that can naturally take advantage of the opportunities presented by heterogeneous networks and D2D communications. This approach enables easy development, deployment, and management of IoT applications in urban computing environments.

II. IoT Applications in Urban Computing Environments

In the near future, IoT applications will put a huge burden on network and computational resources. As many new applications emerge, system designers will have to deal with designing solutions capable of processing very large volumes of data in a timely fashion.

Transmitting all raw IoT data to the Cloud for processing requires a lot of bandwidth and computational resources, introduces significant latency in the data analysis processes and on the delivery of processed information, and places a considerable strain on communication infrastructures.

This is especially true for the cellular network infrastructure. Despite D2D communications representing an interesting opportunity to mitigate the risks of encountering performance bottlenecks at the evolved NodeB (eNB) base stations [7] [8], the projected large volume of generated data will still put a significant burden on the network. This situation is further exacerbated by the increasing sophistication of mobile devices, which opens the door for even more data collection and processing applications. In particular, networks will have to accommodate larger quantities of data as they pave the way to mobile crowdsensing [9] and even anticipatory mobile computing [10] scenarios.

A promising alternative involves pushing computational resources towards the edge of the network, thus enabling low-level data processing via available devices such as event detection and data aggregation. This "fog computing" approach is a vision that extends the long-time proposed mobile and/or on-demand Cloud deployment [11] and is similar to the notion of Agile Computing [6]. By keeping the processing effort at the edge of the network and favoring peer-to-peer communications, using D2D or WiFi ad-hoc, we avoid conveying huge volumes of IoT data on the cellular infrastructure and to the Cloud, which would negatively impact the network infrastructure.

However, the quick evolution of hardware enables even more agile, ad hoc solutions than current Cloud based Infrastructure- or Platform-as-a-Service approaches. Modern hardware solutions based on neuromorphic processors (such as IBM's True North Chip[1]), hybrid CPU/manycore (such as Adapteva's Parallela board[2]) or CPU/FPGA architectures (such as Xilinx's Zynq-7000 SoC[3]), or ultra-cheap CPUs (such as the new Raspberry Pi Zero) will enable the deployment of energy efficient computing platforms capable of "Cloud-like" data aggregation and processing at the network's outer edges. Leveraging those resources will require more dynamic approaches.

Finally, the adoption of the "Big-Data" approach in IoT-based ecosystems, based on the analysis of each single portion of raw data available, may be susceptible to the "diamonds in the raw data" fallacy [4]. According to this fallacy, with increasingly large volumes of IoT data, proportional amounts of valuable information will emerge. We argue that this assumption might not always hold, and consequently that a significant portion of future IoT applications could be better served by a much more efficient approach based on the accurate analysis of a small and targeted portion of IoT data.

To better illustrate the challenging requirements of IoT applications, their increasing relevance, and their need for new communications and computational paradigms, we introduce a possible near future scenario in which IoT applications are used to support participants and workers at a large Street Music Festival. The Street Music Festival is an event involving a relatively large number of street artists performing in an urban environment, attracting a significant number of people.

Customers attending the festival (henceforth termed *participants*) will be interested in information about performances as well as vending options for both refreshments and merchandise. Likewise, vendors will want to track general participant activity to plan their sales activities. For instance, a food vendor may want to time their cooking activities with the conclusion of a nearby performance.

In order to preserve safety and security of participants, the event will be worked by a team of *Emergency Medical Services (EMS)* personnel and by a police force. EMS personnel will want to track the number of people gathering in each area for both resource allocation, e.g., deployment of water, cooling centers, and staff, and management of emergency requests. Likewise, the *police force* will be interested in tracking the occurrence of criminal incidents, such as drunk and disorderly felonies. Additionally, police will be interested in indicators of potential incidents, such as large vehicles fitting specific profiles that could be used for malicious purposes.

In this scenario, the development and deployment of IoT applications requires the installation and management of communications and computational resources through a highly multidisciplinary effort, approaches inspired by Software Defined Networking (SDN) represent a very promising research direction.

III. SPF Design Overview

To address the issues discussed in Section II, we propose SPF (Sieve, Process, and Forward), an SDN-inspired solution that enables IoT data filtering ("sieve" phase), information

[1] http://www.research.ibm.com/articles/brain-chip.shtml

[2] http://www.adapteva.com/parallella-board/

[3] http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

extraction ("process" phase), and dissemination ("forward" phase).

In SPF, an *IoT application* is simply a collection of related services with the same priority and the same target users. A *service* is a function implemented through the processing of IoT data and dissemination of the resultant information that users can activate on-demand. Examples of services include audio/video analysis, as well as object tracking and counting.

The SPF Controller allows *service providers* (or *managers*) to define IoT applications that provide a set of services. SPF enables the definition of several concurrent applications, each with different services and priority levels. In addition, the SPF Controller receives user service requests and deploys services accordingly. This enables the installation of services that require resources for data analysis and information dissemination only when they are needed.

We envision 3 roles for the stakeholders in the SPF architecture: administrators, service providers, and users. *Administrators* manage the SPF platform by deploying, running, and operating SPF controllers and Programmable IoT Gateways (PIGs), and making them available to service providers. *Service providers* develop IoT applications, deploy them, and take care of their management. Finally, *users* are people who use SPF applications through a client app on their smart devices. In our example, users of the SPF are participants, the EMS, and the police force. We can imagine that the management and service provider roles in this scenario would be played by two corresponding commercial companies.

The SPF architecture, depicted in Fig. 1, is based on an extended version the Open Network Foundation's SDN Architecture, with an Information Processing and Dissemination Plane replacing the Data Plane. The two most important components of SPF are a centralized SPF Controller and a set of PIGs deployed at the 6LoWPAN/Internet edge of the network.

The Application Plane of SPF contains all the IoT applications developed and deployed by service providers. Users running client versions of IoT applications on their

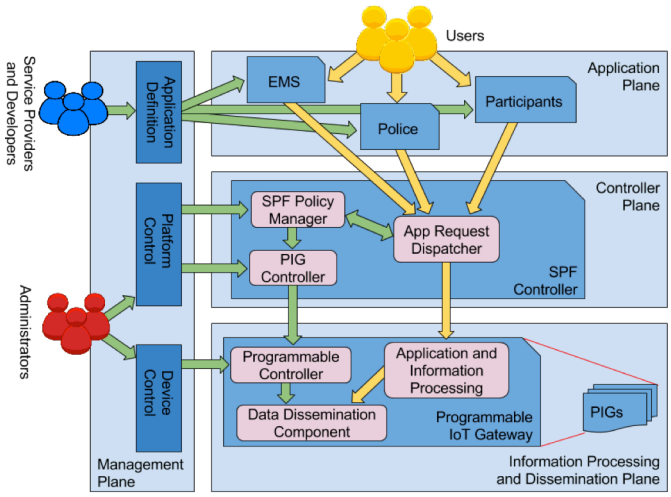

Fig. 1. The SPF Architecture

devices can send service requests to the SPF Controller, typically using the 4G network. The PIGs selected by the SPF Controller will process users' requests and send the produced responses, preferring D2D information dissemination techniques over the use of 4G/LTE.

The functions of the Control Plane are provided by the SPF Controller, which is in charge of deploying the information processing and dissemination functions required by applications. In case an application requests a different service, the SPF Controller reprograms the PIGs accordingly, using 4G/LTE communications. Within the SPF Controller, the Application Request Dispatcher component takes care of receiving service requests from users, of coordinating with the SPF Policy Manager component to identify the most appropriate course of action, and of dispatching the corresponding instructions to the interested PIGs.

The functions of the Information Processing and Dissemination plane are provided by PIGs, which leverage the set of filtering and communications functions implemented by the software platform, according to the instructions received by the SPF Controller. PIGs can be deployed directly on the gateway nodes that connect 6LoWPAN networks to the Internet or on dedicated hardware placed in the gateway nodes' proximity. For simplicity, the rest of this paper will assume the usage of reasonably powerful gateway nodes, each capable of hosting an SPF PIG software component.

Most PIGs will be installed in nodes with multiple communication links: typically Wi-Fi, 4G/LTE, and IEEE 802.15.4, but occasionally also Bluetooth-LE and NFC. The 4G/LTE interface is the only infrastructure-based communications available, allowing (relatively) reliable communications with, e.g., the SPF Controller for reconfiguration purposes. However, 4G/LTE communications are rather resource intensive and are not the preferred solution for battery-operated devices, as many PIGs will be. In fact, SPF enables dissemination of processed information by leveraging short-range device-to-device communications, which, albeit less reliable, represents a solution significantly less resource intensive and particularly well suited for urban computing environments with high node density and node mobility.

The Management Plane provides 3 sets of APIs with independent goals. The Application Definition API enables service providers to define IoT applications and all related configurable parameters. The Platform Control API allows administrators to manage the SPF Controller by giving them the capability to add new PIGs, configure specific information policies or increase/reduce the priority of some applications, and so on. Finally, the Device Control API gives administrators the possibility to control PIGs: this API defines functions to reprogram the number of processing resources allocated for specific services, choose the network interface through which service responses are sent, select the type of information dissemination algorithm to use, and so forth.

IV. IoT Application Definition

A. Definition of IoT Applications and Services

The SPF Controller enables managers to define IoT applications and the corresponding services they implement using a dedicated Domain Specific Language (DSL). Each application has several aspects that can be configured, such as: a *name*, a *priority level* (between 1 and 100), a set of *allowed service types* provided to the users, and a set of *service configurations* and *dissemination policies*. SPF enables IoT applications to receive specific priority levels in order to differentiate between critical and best-effort applications. Service configurations control how the application deals with user service requests of the corresponding type. Dissemination policies instead control all the configurable aspects involved in the information dissemination process, such as: the dissemination channel, the maximum transmission frequency, the maximum number of retransmissions, the obsolescence management policy for the information objects managed by the application, etc.

For instance, in the street music festival scenario from Section II, we could have 3 different IoT applications that support the needs of festival participants, EMS personnel, and police forces respectively.

The *Participants* IoT application only allows 2 types of service requests: "find" and "listen". The find service allows users to look for a specific object, e.g., a text string, in the IoT data. The listen service allows users to request names for songs played in different parts of the festival area, by leveraging a song detection function like those in SoundHound[4] or Shazam[5]. Each of those services also has a service configuration that states that the value of information decays linearly as the distance between user and information source and the time between service request and information generation increase. The Participants applications also has a dissemination policy that defines the information dissemination channel on which information objects will be conveyed to reach their requestors, and the corresponding parameters. The *EMS* and "*Police* applications are defined in similar ways. However, they have higher priorities, no restrictions on the type of services allowed, and more aggressive information dissemination policies.

As with service types, dissemination strategies will vary amongst the 3 applications. A key distinction between the strategies lies in management of object re-transmissions. In the case of the participant application, only one attempt to re-transmit a dropped object is made, after a 30 second wait. By contrast, the police application attempts 60 re-transmissions at 10 second intervals. Additionally, while the Participants and EMS applications will overwrite old objects with newer versions, the Police application also retransmit copies of older object versions (potentially aiding in ongoing investigations).

Another distinction between the dissemination strategies lies in the broadcast channel utilization. In the street music festival scenario, data transmission can be either over local cellular networks or non-cellular WiFi networks set up on the festival grounds. To ensure continuous reliability of local cellular networks, limitations on their usage are taken into consideration. For instance, the Participants application is restricted to WiFi-only transmissions. Likewise, for message re-transmission the EMS application will attempt to use both the cellular and WiFi networks for 1 in 5 transmissions, while the Police application will use both for each transmission. This behavior can be controlled by indicating what percentage of transmissions (by default, 100%) SPF should perform over a particular channel. In general, the careful usage of the many communication resources available represents one of the most important points in an IoT service definition.

B. User Service Requests Management

The SPF Controller receives service requests from users and instantiates the corresponding services accordingly. This allows for dynamic behavior that triggers the execution of information processing and dissemination only when they are actually needed. Since SPF request messages are short, for reasons of simplicity and to avoid increasing latency unnecessarily by employing disruption-tolerant information dissemination techniques, SPF client applications exploit 4G links to send requests to the SPF Controller.

Upon receipt of service requests from users, the SPF Controller updates the state of the service and contacts the PIGs to update their configuration. More specifically, the SPF Controller provides updated information about the number of requests for a given service and the location of the service user closest to the gateway, and changes the configuration of the information dissemination channels in response to a sudden spike (or drop) in the number of service users.

For instance, in the Street Music Festival scenario, Participants are interested in finding information about where they can purchase water and what songs are being played in their proximity. Likewise, EMS personnel are interested in counting the number of participants on the festival grounds, in order to allocate their resources appropriately. Finally, the police force wants to count the number of large vehicles going towards the festival location. Once these requests reach the SPF Controller, they are processed and trigger corresponding reprogramming of the PIGs.

V. Information Processing and Dispatching

The Programmable IoT Gateway (PIG), whose architecture is depicted in Fig. 2, is the component of SPF in charge of information processing and dissemination. The PIG behavior is fully programmable based on the instructions received by the SPF Controller.

A. Dynamic Instantiation of Information Processors

The processing of data is performed by several *information processors* (or *pipelines*) that are set up by the Programmable Controller, according to the instructions received by the SPF Controller. Pipelines, depicted as a cog wheel and funnel sequence in Fig. 2, have the purpose of analyzing raw data to obtain higher-level information, e.g., functions to implement object tracking, object counting, optical character recognition (OCR), and speech-to-text.

[4] SoundHound: http://www.soundhound.com/

[5] Shazam: http://www.shazam.com/

For each message m containing raw data received from the 6LoWPAN, the Information Processor checks which set of services R_S are relevant and tags the message accordingly. Then, the Information Processor detects which set of pipelines R_P are relevant for message m, creates $\overline{R_P} - 1$ duplicates of m, and forwards the messages to the pipelines. For instance, a message containing a video frame would be relevant to the *Find Water* service for the Participants application, to the *Count People* service of the EMS application, and to the *Track Vehicles* service of the Police application. As those services respectively require OCR processing, object counting, and object tracking, copies of the video frame message would be forwarded to the three corresponding pipelines. Note that, in order to minimize the number of information processing tasks, a pipeline performing a specific processing task can be shared between different services.

To minimize the number of information processing tasks, SPF optionally allows implementation of content-wise filtering of the data that is considered for processing, also referred to as the *Sieve* phase. To this end, each pipeline keeps a memory of configurable size (defaulting to 5 messages) that stores the raw data messages recently processed. Before the actual processing, each message is compared with the messages stored in the pipeline memory. Here, σ is defined as the difference between the information contained in the new message and the ones stored in memory. If the amount of new information σ is larger than the pipeline's minimum filtering threshold τ_{FIL}, the message is processed, and is otherwise discarded. For instance, a 0.05 filtering threshold value means that new data will only be considered for processing if it differs at least 5% from data stored in memory.

The *Process* component of SPF is managed via a collection of data-processing pipelines, each instantiated for a particular information processing task, e.g., video processing for object counting, photo processing for OCR, etc. Pipelines will typically be implemented by software. For instance, we are

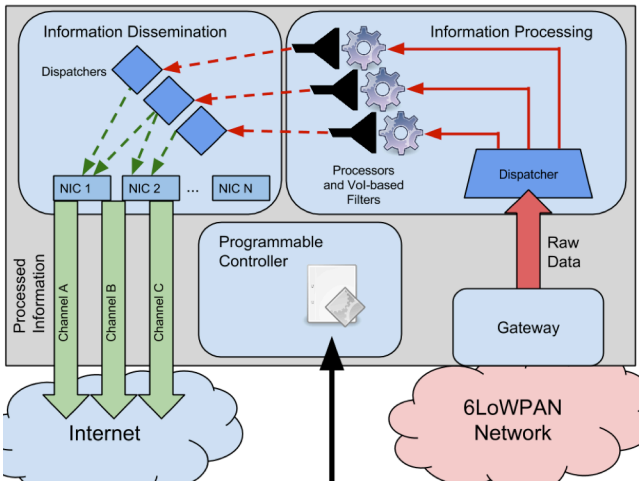

Fig. 2. The SPF Programmable IoT Gateway.

currently implementing basic object recognition / tracking and OCR functions based on open source software including OpenCV [6] and Tesseract [7]. However, pipelines might also leverage existing hardware resources for information processing purposes, such as FPGAs, many-core CPUs, or neuromorphic processors.

After the pipeline processing, the message containing raw data is transformed into an Information Object (IO). Then, the Information Processing component calculates the VoI of the IO for each service $r \in R_S$ using formula (1). It then hands the IO and the corresponding VoI to the Information Dissemination component.

B. *Value of Information-based Prioritization*

To prioritize the dissemination of important information, SPF leverages concepts from Value of Information (VoI) research, which represents one of the most promising avenues for information filtering and prioritization in IoT applications. In fact, by associating a dynamic and recipient-specific value to each Information Object (IO), VoI based methodologies and tools represent a natural way to filter and prioritize information [5].

Multiple definitions exist in the literature for VoI, and it is commonly quantified as the degree of benefit a piece of information provides to decision makers. In [5], the authors define VoI as a metric specific to the needs of particular consumers, as opposed to intrinsic attributes of information termed Quality of Information (QoI). In other words, while an information object may have high intrinsic quality, its value to one consumer may be different than to others. As an example from the Street Music Festival scenario, consider a high-quality (e.g., high resolution) image of a van with tinted windows pulling up near a performance stage. While this image may have low value to participants, it could have a much higher value for police and security forces. Since the quality of this image can impact its ultimate value, [5] also considers QoI as a key part of establishing VoI. By exploiting VoI, SPF can transmit information to appropriate consumers while regulating resource consumption at the edge of the network.

SPF calculates VoI according to four factors: Application Priority (P_R), Normalized Number of Requests (R_N), Timeliness Relevance (of Request) Decay (T_{RD}), and Proximity Relevance (of Request) Decay (P_{RD}).

Each IoT application in SPF is assigned a priority P_R. This is used to establish an ordering in which corresponding information should be sent through the network, independently from others VoI factors.

We expect that the size of consumer populations will vary across different IoT applications. Therefore, SPF calculates the R_N factor to normalize the VoI of each Information Object (IO) based on the size of their respective consumer populations.

The SPF model decreases the VoI of IOs over time using the obsolescence profile modifier, which affects the value of T_{RD}. SPF supports three types of VoI obsolescence profiles: constant, linear decay, and exponential decay. In the constant profile, the VoI of an IO remains constant over time (T_{RD}

[6] http://opencv.org/

[7] https://github.com/tesseract-ocr/tesseract

always equals 1). In the linear decay profile, VoI decreases from a maximum value of 100% ($T_{RD} = 1$) at the time of information generation t_0, to a value of 0% ($T_{RD} = 0$) at time $t = t_0 + \tau_{decay}$. In general, most applications will want to use the linear or exponential VoI profile modifiers, as the constant VoI profile is designed for the most critical IOs only.

SPF also takes into consideration the geographic distance between a consumer and the location corresponding to an information object to compute VoI. In fact, distance values can impact potential usefulness to consumers - knowledge of a vendor on the opposite side of the concert grounds may not be useful, while knowledge of a closer vendor could be. Therefore, an obsolescence profile modifier similar to that used to compute T_{RD} is applied to proximity, which determines the value of factor P_{RD}. Again, linear and exponential VoI profile modifiers are expected to be relevant to most tasks, while the constant modifier will only apply to the most critical IOs.

Based on the four proposed factors, SPF's corresponding VoI calculation formula is defined as:

$$VoI(o,r,t,a) = P_A(a) * R_N(r) * T_{RD}(t, O_T(o)) * P_{RD}(O_L(r), O_L(o)) \quad (1)$$

where o is an information object, r the recipient, t the current time, a the application, O_T and O_L are operators that retrieve, respectively, the time and location of origin of objects and recipients.

C. Information Delivery over Dissemination Channels

Once they are ready for dissemination, IOs are transferred to corresponding *service dispatchers*. Service dispatchers are the entities in charge of the information dissemination process for a specific service. Each service dispatcher maintains a transmission queue, where IOs are ordered according to their corresponding VoI score. IOs with higher VoI scores are transmitted first and those with lower VoI scores are transmitted later.

Infrastructure-based communications will typically be available through 4G/LTE connectivity. However, in addition to their relatively intensive energy consumption, cellular communications can significantly deteriorate when used by a very large number of consumers. For these reasons, service dispatchers will typically prefer conveying IOs over a variety of short-range and D2D communications. In fact, the way the service dispatchers use these devices represents one of the most important policies for IoT service definition.

For instance, for the Find Water service in the Street Music Festival scenario, SPF could leverage D2D communications (Wi-Fi, Bluetooth, or NFC) to enable the dissemination of IOs reporting the locations where participants could find water without burdening the 4G/LTE infrastructure. Infrastructure-less communications could also enable the dissemination of IOs in locations that cannot be easily reached using infrastructure-based communications, such as subway stations.

From the communications perspective, our approach builds on top of our previous research on information dissemination in opportunistic networking and tactical network environments. More specifically, SPF implements its *Forward* phase by leveraging the information dissemination functions provided by the DisService[8] [12] [6] software package.

DisService implements disruption-tolerant publish-subscribe communications by supporting multiple independent virtual communication channels, called *subscriptions*, and is designed to operate in highly dynamic networking environments. To optimize communications, DisService nodes use aggressive message-caching policies and keep track of each other's contact history. This enables nodes to infer knowledge about their surrounding environment, which can be used to detect and leverage message ferrying nodes [13].

SPF uses a dedicated DisService subscription for each IoT application. The dissemination policies for IoT services can also be individually tuned with several parameters, such as the aggressiveness in IO retransmissions and copying, and the aggressiveness in caching usage.

D. Programmable controller

The Programmable Controller is the component responsible for setting up the information processing pipelines, the service dispatchers, and the information dissemination channels according to the instructions received by the SPF Controller.

The Programmable Controller listens on the 4G/LTE interface and waits for a program or service configuration update from the SPF Controller. In fact, during the execution of IoT applications and services, the Programmable Controller can (and typically will) receive additional requests for a new service installation or for the reconfiguration of an existing service, e.g., in case a much larger pool of users ends up using that service. In this case, the Programmable Controller will configure the information processing and dissemination pipelines according to the instructions received from the SPF Controller, instantiating processing components if necessary.

The Programmable Controller is also in charge of managing computational resources and of activating the hardware and software components required for the information processing. For instance, for the *Find Water* service presented in the Street Music Festival scenario, the Programmable Controller is capable of understanding that the object of the find command is a textual string. So, it will instantiate an OCR software component to analyze messages containing pictures and extract any text they contain. Similarly, for an object tracking service the Programmable Controller will instantiate a video processing software component that will try to detect large vehicles coming towards the festival grounds.

VI. Implementation and First Evaluation

We have developed a prototype implementation of SPF using the JRuby platform and have released it as open source under the MIT license. The prototype is available for download at: `https://github.com/mtortonesi/spf`.

We have also developed an Android application that implements a prototype client version of the Participants application in the Street Music Festival scenario. Our application allows the user to send "find"-type requests to a remote SPF Controller, automatically enriched with

[8] DisService is open source: https://github.com/ihmc/nomads

geolocation data. Information on the device position is used for VoI computations and to improve the performance of particular information dissemination strategies. The application interfaces with the DisService application for Android to receive IOs sent by PIGs over DisService.

We also implemented 2 image processing pipelines: *Car Count*, which counts the number of vehicles appearing in an image using the Haar Cascade Classifier algorithm, and *OCR*, which identifies the portions of containing text, cuts them, and extrapolates the text contained in them. Both pipelines are implemented in Java leveraging the functions provided by the OpenCV 3.0 library and the Tesseract and Tess4J components.

We conducted a small experiment to assess the effectiveness of SPF in managing IoT information object throughput, using a PIG on a machine with Ubuntu Linux 15.10 64bit, equipped with an Intel Core 2 Duo P8400 CPU at 2.26GHz and 4GB of RAM, and running the Java 7 HotSpot 64-Bit Server VM. Table 1 shows some preliminary results with respect to the average image processing time obtained for the Car Count and OCR pipelines, using pools of 2, 4, and 8 threads. For the experiment, we fed each pipeline with a dataset of 100 and 64 images for Car Count and OCR, respectively, and each experiment was repeated three times for every possible value of the thread pool size. In addition, the last two columns of the table show the amount of data at the entry and exit points of the pipelines. A difference of three orders of magnitude between input and output size emerges from our experiments, which led us to believe that moving data analysis and feature extraction from the Cloud to the edge of the network can be incredibly effective in reducing strain on the network infrastructure.

TABLE I. AVERAGE PROCESSING TIME PER IMAGE WITH DIFFERENT THREAD POOL SIZES (2, 4, AND 8) FOR THE CAR COUNT AND OCR PIPELINES

	Avg. pr. t. p.i. (ms) p.t.s. 2	**Avg. pr. t. p.i. (ms) p.t.s. 4**	**Avg. pr. t. p.i. (ms) p.t.s. 8**	**Input Size (MB)**	**Output Size (KB)**
Car Count	311.05	286.34	257.41	11.3	8.2
OCR	958.31	934.70	997.224	13.5	19.3

VII. RELATED WORK

A significant amount of research has focused on the problem of data reduction for IoT/M2M applications. Researchers have proposed sophisticated information theoretic and data centric solutions that enable the collection of a subset of IoT data with high entropy levels [14]. In addition, many studies have addressed the more specific problem of data reduction for time series, which are arguably the most widely used data structure in IoT applications [3] [15] [16] [17].

The Quality- and Value-of-Information concepts are relatively novel and have been recently proposed and investigated in the sensor network research area, following the seminal work by Howard [18] and more recent developments in economic and decision theories [19] [20]. More specifically, researchers have developed system-wide i.e., non-consumer specific, and time-invariant QoI- and VoI-based data reduction solutions leveraging multiple-criteria decision making techniques such as the Analytic Hierarchy Process [21] and Von Neumann-Morgenstern utility functions [22]. Other proposals devised more sophisticated schemes that consider time-varying properties in VoI metrics to optimize the scheduling of message transmissions [23] or the traveling path of unmanned data harvesters [24] in underwater wireless sensor networks. To the best of our knowledge, the investigation of time-varying and consumer-specific VoI metrics for dynamic information filtering and prioritization in tactical networks has only been recently investigated in [5].

The fog computing and mobile edge computing research areas also focus on the deployment and exploitation of computational resources at the edge of the network, in proximity to the data sources and to the service consumers [11] [25]. While they represent very promising research areas, fog computing and mobile edge computing appear to be focused on the architectural level and on the mechanisms to realize dynamic allocation of virtual resources. They do not place significant attention on defining and supporting new paradigms for IoT application realization, and are more interested in extending traditional cloud concepts so that they could be used to perform computation at the edge of the network.

SDN-based approaches, while very successful in wired network environments [26] [27], have received a limited attention for IoT/M2M applications. Valdivieso Caraguay et al. [28] provides an interesting review of the opportunities and challenges related to the application of SDN technologies to IoT. Da Silva et al. [29] applied SDN concepts to SCADA networks in order to constrain the potential eavesdropping of critical information. MINA [30] represents one of the first comprehensive SDN-based solutions for IoT applications, but it focuses mostly on low-level aspects such as flow scheduling and resource allocation and it does not consider D2D communications.

Differing from related work that deals with specific aspects of enabling IoT applications, SPF adopts a more holistic approach by considering the application developers perspective and introducing an innovative paradigm for IoT application definition and management. In addition, we note that the information processing solutions adopted by SPF go well beyond the ones currently proposed in the literature in enabling and supporting subjective (i.e., recipient-specific) information filtering and prioritization through VoI concepts. More specifically, this enables SPF to define prioritized application classes by specifying the rules to adopt in assessing the corresponding VoI.

Finally, a considerable amount of research has been dedicated to non-trivial event detection in IoT networks using complex event processing [31] [32] and sophisticated semantic technologies [33]. These efforts are orthogonal and complementary to SPF, which instead adopts an architectural level approach and focuses on enabling and supporting the definition and deployment of IoT applications. Integration of complex event processing functions into SPF would be a very interesting research direction, and is left for future work.

VIII. CONCLUSIONS AND FUTURE WORK

The approach presented in this paper demonstrates that, paired with information processors deployed at the Internet/IoT

edge and VoI-based disruption tolerant information dissemination solutions, SDN represents a very promising architecture for future urban computing applications.

As we believe this research direction is worth further attention, we are going to extend SPF to also consider information generated by mobile devices of consumers. This would enable the development of interesting functions, e.g., the subscription to "trending" thematic information à la Twitter.

We are also planning to investigate distributed and disruption tolerant architectures for the SPF Controller. In fact, the controller is the only centralized component in the SPF architecture, and raises single point of failure and performance bottleneck concerns. Finally, we will further develop the information processing functions of PIGs using both complex event processing and semantic methodologies and tools.

REFERENCES

[1] Cisco Systems Inc., "Cisco Global Cloud Index: Forecast and Methodology, 2013–2018", 2014, available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html (retrieved on December 1st, 2015).

[2] Y. Zheng et al., "Urban Computing: Concepts, Methodologies, and Applications", *ACM Transactions on Intelligent Systems and Technologies*, Vol. 5, No. 3, Article 38, September 2014.

[3] A. Papageorgiou, B. Cheng, E. Kovacs, "Real-Time Data Reduction at the Network Edge ofInternet-of-Things Systems", in *Proceedings of 11th International Conference on Network and Service Management (CNSM)*, 2015.

[4] N. Silver, "The Signal and the Noise: Why So Many Predictions Fail - But Some Don't", Penguin Press, 2012.

[5] N. Suri et al., "Exploring Value of Information-based Approaches to Support Effective Communications in Tactical Networks", *IEEE Communications Magazine*, Vol. 53, No. 10 (Special Feature on Military Communications), October 2015.

[6] G. Benincasa et al., "Agile Communication Middleware for Next-generation Mobile Heterogeneous Networks", *IEEE Software*, Vol. 31, No. 2 (Special Issue on Next Generation Mobile Computing), pp. 54-61, March-April 2014.

[7] A. Asadi, Q. Wang, V. Mancuso, "A Survey on Device-to-Device Communication in Cellular Networks", *IEEE Communications Surveys & Tutorials*, Vol.16, No.4, pp.1801-1819, Fourth quarter 2014.

[8] X. Lin, J. Andrews, A. Ghosh, R. Ratasuk, "An overview of 3GPP device-to-device proximity services", *IEEE Communications Magazine*, Vol.52, No.4, pp.40-48, April 2014.

[9] G. Cardone, A. Corradi, L. Foschini, R. Ianniello, "ParticipAct: a Large-Scale Crowdsensing Platform", to appear in *IEEE Transactions on Emerging Topics in Computing*, 2015.

[10] V. Pejovic, M. Musolesi, "Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges", *ACM Computing Surveys*, Vol. 47, No. 3, Article 47, April 2015.

[11] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog computing and its role in the internet of things", in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing (MCC '12)*, New York, NY, USA, pp. 13-16.

[12] N. Suri et al., "Peer-to-Peer Communications for Tactical Environments: Observations, Requirements, and Experiences", *IEEE Communications Magazine*, Vol. 48, No. 10 (Special Feature on Military Communications), pp. 60-69, October 2010.

[13] M. Marchini et al., "Predicting Peer Interactions for Opportunistic Information Dissemination Protocols", in *Proceedings of the 17th IEEE Symposium on Computers and Communication (ISCC 2012)*, 1-4 July 2012, Cappadocia, Turkey.

[14] H.-Y. Hsieh, C.-H. Chang, W.-C. Liao, "Not Every Bit Counts: Data-Centric Resource Allocation for Correlated Data Gathering in Machine-to-Machine Wireless Networks", *ACM Transactions on Sensor Networks*, Vol. 11, No. 2, Article 38, March 2015.

[15] F. Chung, T. Fu, R. Luk, V. Ng, "Flexible time series pattern matching based on Perceptually Important Points", in *Proceedings of International Joint Conference on Artificial Intelligence, Workshop on Learning from Temporal and Spatial Data*, pp. 1–7, 2001.

[16] K. B. Pratt, E. Fink, "Search for Patterns in Compressed Time Series", *International Journal of Image and Graphics*, Vol. 2, No. 1, pp. 89–106, 2002.

[17] B.-K. Yi, C. Faloutsos, "Fast Time Sequence Indexing for Arbitrary Lp Norms", in *Proceedings of the 26th International Conference on Very Large Data Bases, VLDB '00*, pp. 385–394, 2000.

[18] R. Howard, "Information Value Theory", *IEEE Transactions on Systems Science and Cybernetics,* Vol.2, No.1, pp.22-26, Aug. 1966.

[19] S. Galanis, "The Value of Information under Unawareness", *Journal of Economic Theory*, Vol. 157, pp- 384-396, May 2015.

[20] J. Quiggin, "The Value of Information and the Value of Awareness", to appear in *Theory and Decision*, 2015, DOI: 10.1007/s11238-015-9496-x.

[21] C. Bisdikian, L. Kaplan, M. Srivastava, "On the quality and value of information in sensor networks", *ACM Transactions on Sensor Networks*, Vol. 9, No. 4, Article 48, pp. 48:1-48:26, July 2013.

[22] D. Cansever, "Value of Information", in *Proceedings of 2013 Military Communications Conference (MILCOM 2013)*, pp. 1105-1108, San Diego, CA, USA, 18-20 November 2013.

[23] L. Bölöni, D. Turgut, S. Basagni, C. Petrioli, "Scheduling Data Transmissions of Underwater Sensor Nodes for Maximizing Value of Information", in *Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM 2013),* pp. 460-465, 9-13 December 2013.

[24] S. Basagni et al., "Maximizing the Value of Sensed Information in Underwater Wireless Sensor Networks via an Autonomous Underwater Vehicle", in *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM 2014)*, Toronto, Canada, 27 April – 2 May, 2014.

[25] European Telecommunications Standards Institute, "Mobile-Edge Computing – Introductory Technical White Paper", ETSI Technical Report, September 2014.

[26] A. Lara, A. Kolasani, B. Ramamurthy, "Network Innovation using OpenFlow: A Survey", *IEEE Communications Surveys & Tutorials,* Vol.16, No.1, pp.493-512, First Quarter 2014.

[27] D. Kreutz et al., "Software-Defined Networking: A Comprehensive Survey", *Proceedings of the IEEE*, Vol. 103, No.1, pp.14-76, Jan. 2015.

[28] Á. L. Valdivieso Caraguay, A. B. Peral, L. I. Barona López, L. J. García Villalba, "SDN: Evolution and Opportunities in the Development IoT Applications", *International Journal of Distributed Sensor Networks*, Vol. 2014, Article ID 735142, 10 pages, 2014.

[29] E. G. da Silva et al., "Capitalizing on SDN-based SCADA systems: An anti-eavesdropping case-study", in *Proceedings of 14th IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*, Ottawa, Canada, 11-15 May 2015.

[30] Q. Zhijing et al., "A Software Defined Networking architecture for the Internet-of-Things", in *Proceedings of 14th IEEE/IFIP Network Operations and Management Symposium (NOMS 2014) – Special Session on IoT Management*, Krakow, Poland, 5-9 May 2014.

[31] R. Mayer, B. Koldehofe, K. Rothermel, "Predictable Low-Latency Event Detection With Parallel Complex Event Processing", *IEEE Internet of Things Journal*, Vol. 2, No. 4, pp. 274-286, August 2015.

[32] G. Cugola, A. Margara, "Processing flows of information: From data stream to complex event processing", *ACM Computing Surveys*, Vol. 44, No. 3, Article 15, June 2012.

[33] S. Hasan, E. Curry, "Approximate Semantic Matching of Events for the Internet of Things", *ACM Transactions on Internet Technology*, Vol. 14, No. 1, Article 2, August 2014