Network Function Computation as a Service in Future 5G Machine Type Communications

Dejan Vukobratovic*, Dusan Jakovetic†, Vitaly Skachek‡, Dragana Bajovic§, Dino Sejdinovic¶
*Faculty of Technical Sciences, University of Novi Sad, Serbia, e-mail: dejanv@uns.ac.rs.
†BioSense Institute and Faculty of Sciences, University of Novi Sad, Serbia, email: djakovet@uns.ac.rs.
‡Institute of Computer Science, University of Tartu, Estonia, email: vitaly.skachek@ut.ee.
§BioSense Institute and Faculty of Technical Sciences, University of Novi Sad, Serbia, email: dbajovic@uns.ac.rs.
¶Department of Statistics, University of Oxford, UK, email: dino.sejdinovic@stats.ox.ac.uk.

***Abstract*—The 3GPP machine type communications (MTC) service is expected to contribute a dominant share of the IoT traffic via the upcoming fifth generation (5G) mobile cellular systems. MTC has ambition to connect billions of devices to communicate their data to MTC applications for further processing and data analysis. However, for majority of the applications, collecting all the MTC generated data is inefficient as the data is typically fed into application-dependent functions whose outputs determine the application actions. In this paper, we present a novel MTC architecture that, instead of collecting raw large-volume MTC data, offers the network function computation (NFC) as a service. For a given application demand (function to be computed), different modules (atomic nodes) of the communication infrastructure are orchestrated into a (reconfigurable) directed network topology, and each module is assigned an appropriately defined (reconfigurable) atomic function over the input data, such that the desired global network function is evaluated over the MTC data and a requested MTC-NFC service is delivered. We detail practical viability of incorporating MTC-NFC within the existing 3GPP architecture relying on emerging concepts of Network Function Virtualization and Software Defined Networking. Finally, throughout the paper, we point to the theoretical foundations that inspired the presented architecture highlighting challenges and future directions for designing 3GPP MTC-NFC service.**

***Index Terms*—Internet of Things (IoT), Big Data, Network Coding, Network Function Computation, Machine learning.**

I. Introduction

Internet of Things (IoT) will connect billions of smart devices that will collectively generate and upload a deluge of data to the cloud. By 2020, more than 20 billion of devices will cumulatively generate the IoT data volume exceeding 4.4 ZB (zettabytes) amounting to 10% of the global "digital universe" [1]. The 3GPP machine type communications (MTC) service is expected to contribute a dominant share of this traffic via the upcoming fifth generation (5G) mobile cellular systems [2], [3]. However, a critical bottleneck for the future MTC services is the pressure it puts on the existing communication infrastructure, requiring transfer and storage of enormous data volumes.

Within current IoT/cloud integration, communication infrastructure serves merely to transfer massive raw data to the cloud where applications based on machine learning algorithms extract useful knowledge. However, for majority of services, such an approach is inefficient and wasteful, since the dimensionality of the collected data is typically much smaller than the ambient data space. Thus the approach where communications and data analysis are separated may eventually become unsustainable, calling for a fundamental redesign of IoT communications.

In this paper, we present a generic and reconfigurable MTC architecture, capable of adapting the MTC service to the subsequent data analysis. The new MTC service is based on a common generic interface between data communication and data analysis: *the function computation*. Thus instead of collecting raw and high-volume MTC data, the presented architecture offers the network function computation (NFC) as a service. For a given MTC application (that defines a function to be computed), different modules (atomic nodes) of the involved communication infrastructure (e.g., MTC devices, base stations, and gateways) are orchestrated into a (reconfigurable) directed network topology, and each module is assigned an appropriately defined (reconfigurable) atomic function over the incoming packets. Ultimately, the desired NFC service over the MTC-generated data is delivered at the data center through the composition of the atomic functions.

From the theoretical side, the proposed MTC-NFC service is inspired by recent works that generalize network coding to the concept of network function computation [4], [5]. From the implementation side, we detail a practical viability of incorporating MTC-NFC into the existing 3GPP architecture relying on emerging concepts of Software Defined Networking (SDN) and Network Function Virtualization (NFV) [6], [7]. As an example, we present random linear network coding (RLNC) as an MTC-NFC service [8], however, many other examples can be easily integrated in MTC-NFC (see an extended version of this paper [9]).

The paper is organized as follows. In Sec. II, we present the transition between the layered architecture of current 3GPP MTC service on the one hand, and the new MTC-NFC architecture on the other hand. In Sec. III, we describe the atomic function computation layer that defines the basic building block of the MTC-NFC architecture. Atomic modules are organized into the network function computation layer, as detailed in Sec. IV. An example of the application layer in the form of a RLNC-based MTC-NFC service is discussed in Sec. V. Sec. VI concludes the work.

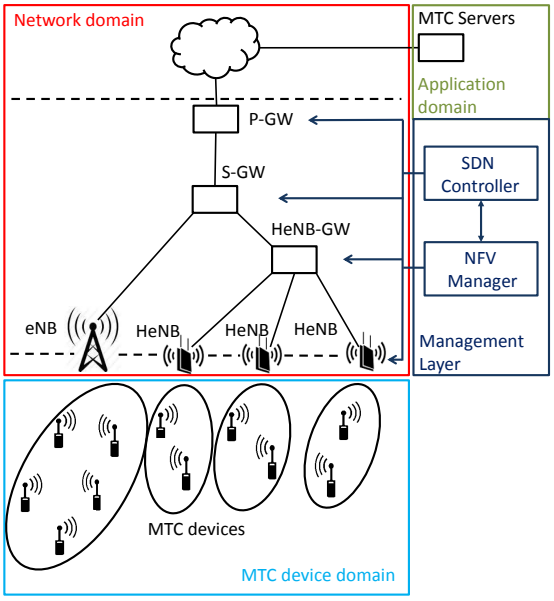

Fig. 1. The 3GPP MTC architecture.

II. The MTC-NFC Service Architecture

A. The 3GPP MTC Architecture

European Telecommunications Standards Institute (ETSI) initiated the work on Machine Type Communications (MTC or M2M) architecture that would serve to connect a massive amount of devices to wireless mobile cellular networks. MTC is designed to enable MTC devices to send data to each other or to a set of MTC servers [2]. Specific parts of MTC standardization efforts are addressed by 3GPP standardization as of Release 10 standards and beyond [3].

Fig. 1 illustrates the 3GPP MTC architecture. Abstracted to its essence, the current 3GPP MTC architecture is represented by three layers. The *data layer*, formally named the MTC device domain, contains billions of MTC devices that generate data while interacting with the environment. The *communication layer*, or formally called the network domain, merely transfers data by essentially uploading all the captured data to MTC servers (e.g., in the cloud). Finally, the *application layer*, or the MTC application domain, contains data centres running MTC servers which provide storage and processing capabilities.

Focusing on the data plane, the figure illustrates the data path that starts at MTC devices and proceeds via Radio Access Network (RAN) elements: base stations (eNB) or small cells (HeNB's), Evolved Packet Core (EPC) elements: HeNB gateways (He-GW), service gateways (S-GW) and packet gateways (P-GW), finishing at MTC servers. For many applications, cloud-based MTC servers will use machine learning algorithms to extract actionable knowledge from the collected data.

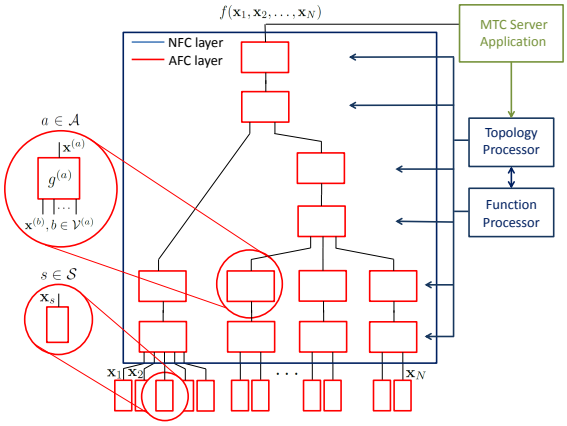

Fig. 2. MTC-NFC architecture.

B. MTC-NFC Architecture: Modules and Layers

In the following, we present a novel MTC architecture that upgrades 3GPP MTC with the concept of network function computation (NFC) [9]. Instead of communicating raw MTC device data, the novel MTC-NFC service delivers function computations over the data. We present the layered MTC-NFC architecture that is generic, flexible and reconfigurable, and is designed to meet the needs of an increasing number of MTC applications that extract knowledge from data avoiding wasteful raw data collection.

The MTC-NFC architecture is presented in Fig. 2. It consists of an interconnected set of basic modules called atomic function computation (AFC) modules that jointly comprise *the AFC layer*. AFC modules evaluate basic (atomic) functions over the input data packets and deliver function evaluations as the output data packets. The collection of interconnected and jointly orchestrated AFC modules represent *the NFC layer* that delivers a network function computation over the source data packets. The resulting NFC evaluations are the output of the MTC-NFC service offered to the MTC servers at *the application layer*.

More formally, we consider an MTC network containing N MTC devices representing the set of source nodes (modules) $\mathcal{S}$. A source node $s \in \mathcal{S}$ produces a packet $\mathbf{x}_s = (x_s[1], x_s[2], \ldots, x_s[L])$ containing L symbols from alphabet $\mathbb{A}$ at the output interface (for simplicity, we assume a single output interface). The MTC network also contains the set $\mathcal{A}$ of M AFC nodes (modules). An AFC node $a \in \mathcal{A}$ receives the set of input data packets $\{\mathbf{x}^{(b)}\}_{b \in \mathcal{V}^{(a)}}$ on its input interfaces, which receive data packets from the set $\mathcal{V}^{(a)}$ of AFC nodes. At the output interface, the AFC node a delivers the output data packet $\mathbf{x}^{(a)}$ (for simplicity, we assume a single output interface). AFC node a associates an atomic function $g^{(a)}$ to the output interface, where $\mathbf{x}^{(a)} = g^{(a)}(\{\mathbf{x}^{(b)}\}_{b \in \mathcal{V}^{(a)}})$. Finally, the MTC network contains R MTC servers as the set of destination nodes $\mathcal{D}$.

The source nodes $\mathcal{S}$, AFC nodes $\mathcal{A}$ and destination nodes $\mathcal{D}$ jointly constitute an NFC graph $\mathcal{G} = (\mathcal{V} = \mathcal{S} \cup \mathcal{A} \cup \mathcal{D}, \mathcal{E})$,

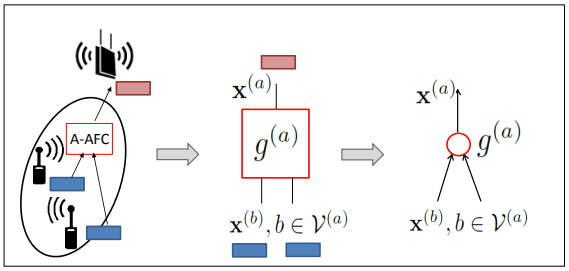

Fig. 3. Wireless-domain A-AFC module.

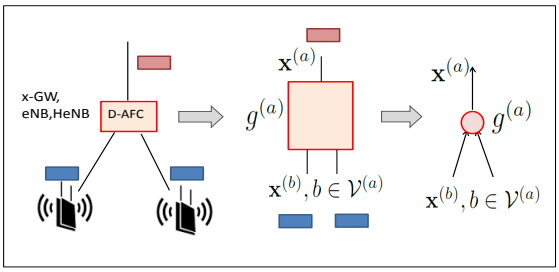

Fig. 4. D-AFC module.

where $\mathcal{V}$ is the set of nodes (modules) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (connections between modules). In the case of directed rooted trees or directed acyclic graphs, the collection of sets $\{\mathcal{V}^{(v)}\}_{v \in \mathcal{V}}$ fully describes the set of connections between modules in the NFC graph (see Fig. 5).

Finally, we introduce the control elements: *topology processor* and *function processor*. Based on the MTC server application requirements, the function processor decomposes a required network function $f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ into a composition of local atomic functions $\{g^{(a)}\}_{a \in \mathcal{A}}$ and configures each AFC module accordingly. The topology processor interconnects AFC modules into a directed NFC graph of MTC data flows by defining the set $\{\mathcal{V}^{(v)}\}_{v \in \mathcal{V}}$. Therefore, the topology and function processors are the NFC layer entities that manage, connect and orchestrate the AFC layer entities (source and AFC modules). As we will explain ahead, the two processors can be naturally implemented using the concepts of SDN and NFV.

III. Atomic Function Computation Layer

The AFC layer is composed of AFC modules that evaluate atomic functions such as addition, modulo addition, maximum/minimum, norm, histogram, linear combination, threshold functions, etc. We consider two types of AFC modules: i) Analog-domain AFC (A-AFC), and ii) Digital-domain AFC (D-AFC) modules. The former exploit superposition of signals in the analog domain, while the latter evaluate atomic functions straightforwardly in the digital domain using digital processing in network nodes.

A. Analog-domain Atomic Function Computation (A-AFC)

An A-AFC harness interference in a wireless channel or signal combining in an optical channel to perform atomic function evaluations. An example of the technology that can be easily integrated as an A-AFC module is the Physical Layer Network Coding (PLNC) [10], where the corresponding atomic function is the finite field addition. More recently, the concept of compute-and-forward (CF) has been proposed, that also fits well as an A-AFC technology. Therein, linear combinations of input symbols are generated as the output [11]. Finally, in [12], non-linear function computation over wireless channels is addressed, where more general, non-linear functions are computed through introducing a non-linear preprocessing and post-processing of packets before and after the signals have been superimposed in the channel. Fig. 3 illustrates A-AFC: its position in the real-world system (left), its representation as an A-AFC module (central), and as part of the NFC graph (right).

Although above examples demonstrate that A-AFC modules are viable technology, current research is limited in terms of the computed functions, and mostly targets wireless (and not optical) channels. Furthermore, design and implementation of generic A-AFC in wireless setting adaptive to the channel impairments (e.g., estimation errors, timing and frequency offsets, quantization issues, etc.) represents a challenging problem.

B. Digital-domain Atomic Function Computation (D-AFC)

D-AFC modules evaluate atomic functions in the digital domain. This is done within the network nodes such as base stations (eNB or HeNB) and core network gateways (HeNB-GW, S-GW, P-GW). Although digital-domain in-node processing offers many possibilities for D-AFC implementation, we address here two options.

The first option are reconfigurable hardware-based Field Programmable Gate Array (FPGA) platforms frequently used in combination with high-speed networking equipment for high-throughput processing over data packets. FPGAs offer flexible and reconfigurable high-throughput implementations of linear or non-linear atomic functions. For example, FPGA implementation of random linear combinations of incoming data packets, as part of RLNC in network nodes, is recently considered in [13].

The second possibility for D-AFC is to use software-based implementations in high-level programming languages that run within general processing units, either in network nodes or externally on dedicated general-purpose servers [14]. This approach offers full flexibility for atomic function evaluation for the price of lower processing throughput, as compared to the FPGA approach. For example, software-based D-AFC implementation of random linear combinations over incoming packets, as part of RLNC, is available at [15].

Fig. 4 illustrates D-AFC: its position in the real-world system (left), its representation as an D-AFC module (center), and as part of the NFC graph (right).

IV. Network Function Computation Layer

In this section, we review recent fundamental results on network function computation that underlie the NFC layer [4], [5]. Then, we describe the implementation framework using the emerging SDN/NFV concepts.

A. Theoretical Aspects of NFC Layer

Consider a finite directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consisting of M AFC nodes in $\mathcal{A}$, N sources in $\mathcal{S}$, and R destinations in $\mathcal{D}$, such that $\mathcal{S} \cap \mathcal{D} = \varnothing$. The network uses a finite alphabet $\mathbb{A}$, called *network alphabet*. Each source s generates K random symbols $\sigma_s[1], \sigma_s[2], \ldots, \sigma_s[K] \in \mathbb{A}$. Here, we say that the source symbol $\sigma_s[k]$ belongs to the k-th generation of the source symbols.

Each packet sent over a network link is a vector of length L over $\mathbb{A}$. Suppose that each of the R destination nodes requests computation of a (vector-valued) function f of the incoming MTC device vectors $\boldsymbol{\sigma}_s$, $s = 1, ..., N$. The target vector function is of the form $f : \mathbb{A}^{N \cdot K} \to \mathbb{B}^K$, where $\mathbb{B}$ is a function alphabet, and each component function $f : \mathbb{A}^N \to \mathbb{B}$ is of the same form, applied to each source's k-th symbol, $k = 1, ..., K$. More precisely, we wish to compute $f(\sigma_1[k], ..., \sigma_N[k])$, $k = 1, ..., K$.

With each arc $a \to v$ outgoing an AFC node $a \in \mathcal{A}$, we associate the atomic function $g^{(a \to v)}(\cdot)$, which takes the $|\mathcal{V}^{(a)}|$ length-L incoming vectors $\mathbf{x}^{(u \to a)}$, $u \in \mathcal{V}^{(a)}$, and produces the length-L outgoing vector $\mathbf{x}^{(a \to v)}$, i.e.:

$$\mathbf{x}^{(a \to v)} = g^{(a \to v)} \left(\{\mathbf{x}^{(u \to a)}\}_{u \in \mathcal{V}^{(a)}} \right).$$

Similarly, with each arc $s \to v$ outgoing a source node $s \in \mathcal{S}$, the atomic function $g^{(s \to v)}(\cdot)$ takes the $|\mathcal{V}^{(s)}|$ length-L incoming vectors $\mathbf{x}^{(u \to s)}$, $u \in \mathcal{V}^{(s)}$ (allowing input edges to source nodes), as well as the K generated symbols $\boldsymbol{\sigma}_s = (\sigma_s[1], ..., \sigma_s[K])$, and produces the length-L outgoing vector $\mathbf{x}^{(s \to v)}$, i.e.:

$$\mathbf{x}^{(s \to v)} = g^{(s \to v)} \left(\{\mathbf{x}^{(u \to s)}\}_{u \in \mathcal{V}^{(s)}}; \boldsymbol{\sigma}_s \right).$$

We refer to both $g^{(a \to v)}$'s and $g^{(s \to v)}$'s as *encoding functions*.

Finally, a destination node $d \in \mathcal{D}$ takes its $|\mathcal{V}^{(d)}|$ incoming length-L messages and performs decoding, i.e., it produces the vector of function evaluation estimates $\widehat{\mathbf{f}}^{(d)} = \left(\widehat{f}^{(d)}[1], ..., \widehat{f}^{(d)}[K] \right)$, as follows:

$$\widehat{\mathbf{f}}^{(d)} = \Psi^{(d)} \left(\{\mathbf{x}^{(u \to d)}\}_{u \in \mathcal{V}^{(d)}} \right),$$

where $\Psi^{(d)}(\cdot)$ is the destination node d's function. Note that a decoding function $\Psi^{(d)}(\cdot)$ recovers back the K-dimensional vector from the L-dimensional incoming quantities (where $L > K$).

The destination $d \in \mathcal{D}$ *computes* the function $f : \mathbb{A}^N \to \mathbb{B}$, if for every generation $k \in \{1, ..., K\}$, it holds that:

$$\widehat{f}^{(d)}[k] = f(\sigma_1[k], ..., \sigma_N[k]).$$

Further, the problem of computing f is *solvable* if there exist atomic functions $g^{(s \to v)}(\cdot)$, $g^{(a \to v)}(\cdot)$ across all arcs in $\mathcal{E}$ and decoding functions $\Psi^{(d)}(\cdot)$, $d = 1, ..., R$, such that f is computed at all destinations $d \in \mathcal{D}$.

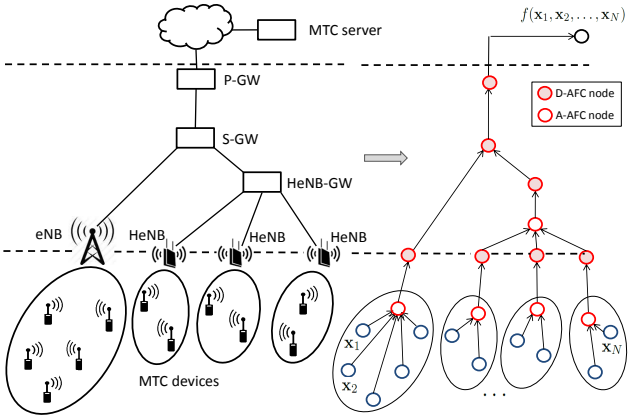

Fig. 5. Mapping between MTC network and NFC graph.

In order to understand the fundamental limits on solvability of the general function computation problem, the authors of [5] define *the computing capacity* of a network as follows

$$\mathsf{C}(\mathcal{G}, f) = \sup \left\{ \frac{K}{L} \ : \ \text{computing } f \text{ in } \mathcal{G} \text{ is solvable} \right\}.$$

They derive a general min-cut type upper bound on the computing capacity, and a number of specific lower bounds. For an extended discussion on NFC, see [9].

B. Implementation Aspects of NFC Layer

The NFC layer can be naturally implemented within the SDN/NFV architecture [6], [7]. In particular, the topology processor (TP) fits as an SDN application running on top of the SDN controller within the SDN architecture. The function processor (FP) role may be set within a NFV manager entity, taking the role of the NFV orchestrator. Using the SDN/NFV framework, MTC-NFC service can be quickly set and flexibly reconfigured according to requests arriving from a diverse set of MTC applications. In other words, the NFC layer should deal with control and management tasks of establishing and maintaining an NFC graph of AFC modules (Fig. 5) for a given service request.

The TP module manages the MTC data flows via the SDN control plane. For all nodes in a directed acyclic graph, the TP provides the set of children nodes $\{\mathcal{V}^{(v)}\}_{v \in \mathcal{V}}$ from which to accept MTC data flows, and identify the exact MTC data flows that will be filtered within the MTC-NFC service.

Based on the MTC server requests and the configured topology, the FP module processes the global function request and generates the set of atomic functions to be used: $\{g^{(a)}\}_{a \in \mathcal{A}}$. Note that, as described before, AFC modules may be: i) A-AFC modules (e.g., PLNC module), ii) hardware-based D-AFC modules (e.g., FPGA module), and ii) software-based D-AFC modules. For example, for the most flexible case of software-based D-AFC modules, a library of AFC implementations could be installed in

network nodes where each atomic function from the library can be remotely instantiated via the NFV concept. We note that a similar approach is recently suggested for RLNC as a service in [8].

V. Application Layer

As a simple example, we consider the MTC-NFC service that delivers random linear combinations in a finite field applied over the MTC device data. We note that this instance of MTC-NFC services may be considered as implementing a special case of RLNC as a service [8]. Although the RLNC example does not reduce the MTC data traffic, i.e., it is equivalent to computing the *identity function* over the input data, it is instructive as it is simple and its components are already available technology. However, it is important to note that the proposed MTC-NFC service targets computation of a wide range of linear and non-linear functions over the input data. We refer the interested reader to [9] for additional examples of minimization of a population risk (statistical estimation/learning of an unknown vector-valued parameter), and non-linear classification via neural networks.

For simplicity, we assume that graph $\mathcal{G}$ is a directed rooted tree. Each of the N MTC devices has a packet $\mathbf{x}_s$ containing L symbols from a finite field $\mathbb{F}$. The goal is to use standard RLNC approach to robustly deliver the whole packet vector $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ to the destination node d [16]. Within the process, each atomic node a generates the message pair $\left(\mathbf{x}^{(a)}, \mathbf{c}^{(a)}\right)$ (to be sent to the parent node) based on the received messages from its children nodes $\left(\mathbf{x}^{(b)}, \mathbf{c}^{(b)}\right)$, where $b \in \mathcal{V}^{(a)}$. The message $\mathbf{x}^{(a)} \in \mathbb{F}^L$ is by construction a linear combination of the input packets $\mathbf{x}_b \in \mathbb{F}^L, b \in \mathcal{V}^{(a)}$. Quantity $\mathbf{c}^{(a)} = (c^{(a)}[1], ..., c^{(a)}[N]) \in \mathbb{F}^N$ is usually called global encoding vector. Once the destination (root) node d receives all its incoming messages, it has available a random linear combination over all the MTC's packets $\mathbf{x}_1, ..., \mathbf{x}_N$: $\mathbf{x}^{(d),1} = \sum_{s=1}^{N} c^{(d),1}[s] \cdot \mathbf{x}_s$ and the corresponding global encoding vector $\mathbf{c}^{(d),1} = \left(c^{(d),1}[1], ..., c^{(d),1}[N]\right)$. The process is repeated sequentially such that the data center obtains N' pairs $\left(\mathbf{x}^{(d),k}, \mathbf{c}^{(d),k}\right)$, $k = 1, 2, ..., N'$. It can be shown that, as long as N' is slightly larger than N, MTC data vector $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ can be recovered with high probability through solving the linear system of equations with unknowns $\mathbf{x}_s$: $\mathbf{x}^{(d),k} = \sum_{s=1}^{N} c^{(d),k}[s]\ \mathbf{x}_s$,$k = 1, ..., N'$. Note that, for this application example, each atomic function is *linear*. Moreover, there is no requirement on the *coordination* of the atomic functions which correspond to different atomic nodes, as they are generated randomly and mutually independently. Hence, this application does not require a centralized control by the function processor. For more involved examples, we refer the interested reader to [9]

Finally, when certain a priori knowledge on $\mathbf{x}$ is available (e.g., sparsity), the recovery probability close to one can be achieved even when the number of linear combinations N' at the MTC server is significantly smaller than N. Omitting details, this could be in principle achieved using the theories of compressed sensing and sparse recovery, e.g., [17].

VI. Conclusions

In this paper, we presented a novel architecture for knowledge acquisition of the MTC data, referred to as Condense [9]. The architecture is designed to provide a generic, flexible and reconfigurable function computation service for MTC applications. The novel MTC-NFC service offers transfer of only the desired function of the MTC-generated data (as required by the given application at hand) – and not the raw data in its entirety – to the data center of interest. This transformational approach has the potential to dramatically reduce the pressure on the 3GPP MTC communication infrastructure.

Acknowledgment

The authors thank G. Karabulut Kurt, C. Hollanti and I. Fischer for further contributions to the concept presented in this work that lead to extended version of this paper [9].

References

[1] http://www.zdnet.com/topic/the-power-of-iot-and-big-data/

[2] T. Taleb and A. Kunz, "Machine type communications in 3GPP networks: Potential, challenges, and solutions," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 178-184, 2012.

[3] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jantti, and A. Ghosh, "Machine-type communications: current status and future perspectives toward 5G systems," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 10-17, Sept. 2015.

[4] H. Kowshik and P.R. Kumar, "Optimal function computation in directed and undirected graphs," *IEEE Transactions on Information Theory,* vol. 58, no. 6, pp. 3407-3418, June 2012.

[5] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 1015–1030, Feb. 2011.

[6] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.

[7] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Comms. Magazine*, vol. 53, no. 2, pp. 90-97, Feb. 2015.

[8] D. Szabo, A. Csoma, P. Megyesi, A. Gulyas, and F. H. Fitzek, "Network Coding as a Service," arXiv preprint arXiv:1601.03201, 2016.

[9] D. Vukobratovic, D. Jakovetic, V. Skachek, D. Bajovic, D. Sejdinovic, G. Karabulut Kurt, C. Hollanti, and I. Fischer: "CONDENSE: A Reconfigurable Knowledge Acquisition Architecture for Future 5G IoT," to appear, *IEEE Access*, 2016.

[10] S. Zhang, S. C. Liew, and P. P. Lam, "Hot topic: physical-layer network coding," *12th annual international conference on Mobile computing and networking*, pp. 358-365, ACM, Sep. 2006.

[11] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463-6486, Oct. 2011.

[12] M. Goldenbaum, H. Boche, and S. Stańczak, "Harnessing interference for analog function computation in wireless sensor networks," *IEEE Trans. Signal Processing*, vol. 61, no. 20, pp. 4893–4906, Oct. 2013.

[13] S. Kim, W. S. Jeong, W. W. Ro, and J. L. Gaudiot, "Design and evaluation of random linear network coding Accelerators on FPGAs," *ACM Trans. Embed. Comp. Systems (TECS)*, vol. 13, no. 1, pp. 13, 2013.

[14] http://steinwurf.com/tag/kodo/

[15] J. Hansen, D. Lucani, J. Krigslund, M. Médard, and F. Fitzek, "Network coded software defined networking: enabling 5G transmission and storage networks," *IEEE Comm. Mag.*, vol. 53, no. 9, pp. 100-107, Sept. 2015.

[16] C. Fragouli and E. Soljanin, "Network coding: Fundamentals and applications," Foundations and Trends in Networking, vol. 2, no. 1, 2007.

[17] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine,* vol. 25, pp. 21-30, March 2008.