**11ème Atelier en Evaluation de Performances**
**Toulouse, du 15 au 17 mars 2016**

# Resource Allocation for in-Network Computations

Apostolos Destounis[1], Georgios S. Paschos[1], Iordanis Koutsopoulos[2]

[1]Mathematical and Algorithmic Sciences Lab,
Huawei technologies FRC
20, quai du Point du Jour
92100 Boulogne-Billancourt, France
firstname.lastname@huawei.com

[2]Department of Informatics, Athens University of Economics and Business (AUEB)
76, Patission Str., 104 34, Athens, Greece
jordan@aueb.gr

## 1. Introduction

In this work, we ask the following question : Given a network graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ with links of limited communication bandwidth and nodes of limited computation resources, what are the *performance limits* of in-network computation throughput ? Namely, what is the *maximum rate* with which computation results can be conveyed to the destination when computations take place in the network ?
We assume there exist two source nodes $s_1, s_2 \in \mathcal{N}$ and a destination node $d$. Edge $(m, l) \in \mathcal{E}$ between nodes $m$ and $l$ has a fixed capacity of $R_{ml}$ packets per slot. A network example is given in Fig. 1.
We study a *stream* of queries, where each query concerns the computation of the sum of a datum from source 1 and a datum from source 2, while the network is agnostic to specificities of data. Upon arrival of each query, a corresponding packet (datum) is generated at each of the two source nodes, and both packets are given the same *tag*. These packets need to be summed somewhere in the network, and the result needs to be delivered to the destination $d$. Time is slotted, and at each slot $t$ there are $A(t)$ newly arrived queries belonging to the same stream, random with $\mathbb{E}[A(t)] = \lambda$.
Combination of packets corresponding to a query may take place in one among a subset of nodes, denoted by $\mathcal{N}_C = \{n_1, n_2, ..., n_{N_C}\} \subseteq \mathcal{V}$ ; these are referred to as the *computation nodes*. Node $n_i$ has computational capacity of $C_{n_i}$, measured in number of produced processed packets per slot, where each processed packet concerns the sum of two raw packets with the same tag when both are available to the computation node.

The objective of this work is to characterize the maximum rate of queries that can be accommodated by the network (referred to as the computation capacity of the network), and provide an online algorithm to achieve this capacity. We restrict ourselves to policies that use only packet routing (i.e. no network coding).

## 2. Queueing Structure

To capture all packet classes in the network we define the following queues (A calligraphic sign denotes a set with the tags of the corresponding packets, normal sign denotes the cardinality of this set) :

— $\mathcal{Q}_k^{(i,n)}(t), i = 1, 2$ : Data queue at node $k$ containing raw packets generated at node $s_i$ that have to be computed at node $n$.
— $\mathcal{X}_n^{(i)}(t), i = 1, 2$ : Computation queue at node $n$ containing raw packets generated at node $s_i$ that have to be computed at *this node*.
— $\mathcal{Q}_k^{(0,n)}(t), i = 1, 2$ : Data queue at node $k$ containing processed packets from computing node $n$, that have to be delivered at the destination node.

In addition, each computation node has queues $\mathcal{Y}_n(t)$ keeping the results of computations (see also Fig. 2) and virtual queues $H_n(t)$ tracking the computation capacity budget.
Moving packets between queues corresponds to control decisions to be taken each slot :

— The set of raw packets originated from node $s_i$, destined to computation node $n$, to be transmitted from node $m$ to node $k$.
— The pairs of raw packets to be combined at each computation node $n$
— The set of processed packets, combined at node $n$, to be transmitted from node $m$ to node $k$.

We have the following constraints : (i)The total number of transmitted packets over a link $(kl)$ are limited by link capacity $R_{kl}$ (ii) the number of combined pairs cannot exceed the computation capacity or any of the individual raw packet queue lengths, and (iii) a pair of packets can be combined only if both packets with the same tag have already arrived at the computation node.

## 3. Upper Bound on the Network Computation Capacity

We define a set $\widetilde{\mathcal{C}}_3$ with $3N_C$ unicast commodities, as follows : there are three commodities for each

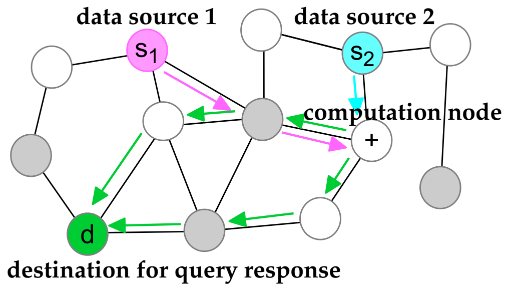

FIGURE 1 – Illustration of network computations. Shaded nodes are forwarding ones, i.e. without computation capabilities, and white nodes have computation capabilities. Arrows denote routing of raw and processed data.

computation node $n \in \mathcal{N}_C$ ; $(1, n)$ delivering packets from $s_1$ to $n$, $(2, n)$ delivering packets from $s_2$ to $n$, and $(n, d)$ delivering combined packets at (computation) node $n$ to the destination. Let $\Lambda(\mathcal{G})$ be the feasible rate region for these commodities on network $\mathcal{G}$. We can characterize the computation capacity as follows :

**Theorem 1** *An upper bound on the computation capacity is given by the following optimization problem :*

$$\lambda^* = \max_{(\lambda_n)} \sum_{n \in \mathcal{N}_C} \lambda_n \tag{1}$$

$$\text{s.t.} \quad 0 \leq \lambda_n \leq C_n, \forall n \in \mathcal{N}_C \tag{2}$$

$$(\lambda_1, \lambda_1, \lambda_1, ..., \lambda_{N_C}, \lambda_{N_C}, \lambda_{N_C}) \in \Lambda(\mathcal{G}) \tag{3}$$

## 4. Algorithm

The dynamic policy we consider here is the following :

1. **Load Balancing :** At each slot, choose $n^*(t)$ equal to

$$\arg \min_{n \in \mathcal{N}_C} \Bigg[ (1 + \epsilon_B) Q_n^{(0,n)}(t) + \sum_{i=1,2} Q_i^{(i,n)}(t) + H_n(t) \Bigg]$$

   where $\epsilon_B \in (0, 1)$ is a control parameter. Then, all newly arrived queries are assigned to the class that corresponds to this computation node.

2. **Routing and scheduling :** Use Backpressure over class pairs. For every link $(m, k) \in \mathcal{E}$ choose the class pair

$$(i^*_{mk}(t), n^*_{mk}(t)) = \arg \max_{i \in \{0,1,2\} n \in \mathcal{N}_C} \left| Q_m^{(i,n)}(t) - Q_k^{(i,n)}(t) \right| .$$

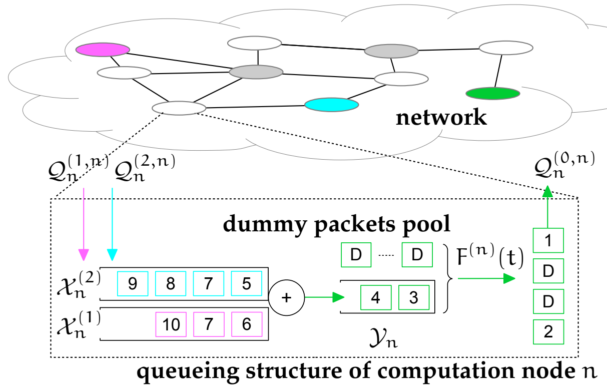

FIGURE 2 – Illustration of queueing structure for computation node $n$. Numbered packets are either raw or processed (green) useful packets, while packets noted with "D" are dummy packets.

   Then use the capacity of the link to route (any) packets of the above class pair, from the biggest to smallest corresponding queue.

3. **Computation :** At every node $n \in \mathcal{N}_C$, all possible computations are done. If there are more pairs than the computation capacity of this node, then $C_n$ pairs are selected using any tie breaking rule.

4. **Randomization with dummy packets :** $F^{(n)}(t) = \mathbb{1}_{\{n=n^*(t)\}} A(t) \left(1 + B^{(n)}(t)\right)$ packets resulting from a computation are pushed to queue $\mathcal{Q}_n^{(0)}(t)$, where $B^{(n)}(t)$ are an i.i.d. Bernoulli random variables with mean $\epsilon_B$. If there are not enough processed packets available at queue $\mathcal{Y}_n$, dummy packets are used.

5. **Update Virtual Queues :**

$$H_n(t + 1) = [H_n(t) - C_n]^+ + \mathbb{1}_{\{n=n^*(t)\}} F^{(n)}(t).$$

The main result of this work is the performance of the online algorithm :

**Theorem 2** *The online policy satisfies any query rate* $\lambda < \left(1 - \frac{\epsilon_B}{1+\epsilon_B}\right) \lambda^*$.

Theorem 2 implies that the online algorithm achieves a computation rate arbitrarily close to the upper bound.