Deterministic Scheduling of Periodic Messages for Cloud RAN

Dominique Barth[1], Maël Guiraud[1,2], Brice Leclerc[2], Olivier Marcé[2], and Yann Strozecki[1]

[1]David Laboratory, UVSQ
[2]Nokia Bell Labs France

June 22, 2018

1 Model and Problems

We use the notation $[n]$ to denote the interval of n integers $\{0,\ldots,n-1\}$.

1.1 Discrete time model

In the model presented here, the time is discrete. The unit of time is called **tic**. It is the time needed to transmit a minimal unit of data over the fastest link of the network.

1.2 Network modeling

The network is modeled as a directed graph $G = (V,A)$. Each arc (u,v) in A is labeled by an integer weight $\Omega(u,v)$ which represents the time taken by a message to go from u to v using this arc, expressed in tics. A **route** r in G is a directed path, that is, a sequence of adjacent vertices $u_0,\ldots,u_l$, with $(u_i,u_{i+1}) \in A$. The **weight of a vertex** u_i in a route $r = (u_0,\ldots,u_l)$ is defined by $\lambda(u_i,r) = \sum_{0\leq j<i} \Omega(u_j,u_{j+1})$. It is the time needed by a message to go from the first vertex of the route to u_i. We also define $\lambda(u_0,r) = 0$. The weight of the route r is defined by $\lambda(r) = \lambda(u_l,r)$. We denote by $\mathcal{R}$ a set of routes, the pair $(G,\mathcal{R})$ is called a **routed network** and represents our telecommunication network. The first vertex of a route models an antenna (RRH) and the last one a data-center (BBU) which computes an answer to the messages sent by the antenna.

1.3 Messages dynamic

In the process we study, some **datagrams** are sent on each route. The size $|d|$ of a datagram d is an integer, expressed in tics. It is the time needed by a node to emit the datagram. Once a datagram have been emitted, it cannot be fragmented during its travel in the network.
Let $r = (u_0, \ldots, u_l)$ be a route, a datagram d_j sent on the route r can be buffered in a node u_i. The network follows the *store and forward* policy, i.e., a datagram is buffered in a node during at least $|d_j|$ tics. The node waits to completely receive the datagram before re-sending it in the network. To each datagram d_j is thus associated to a set of integers $\{b_j^0, \ldots, b_j^l\}$. Those integers, represent the time waited in some **buffers**, corresponding to the buffering time of this datagram in each nodes of the route. Because of the previous remark, $b_j^i \in \{0\} \cup \{|d_j|, \ldots, \infty\}$
We call **offset** of a datagram on a route r, denoted by $o_r(d)$, the time at which a datagram d_j is sent from u_0, the first vertex of r. We say that the tic at which a datagram d_j sent at time $o_r(d_j)$ on r reaches a vertex u_i in r is $t(d_j,u_i,r) = o_r(d_j) + \lambda(u_i,r) + \sum_{k=0}^{i-1} b_j^k$.

A **Slice** S is a sequence of several datagrams. There is one slice for each route of the routed network. The cardinal of a slice, denoted $\overline{S}$ is the number of datagrams in the frame. Let us consider a slice S_i on the route r_i, the datagrams $\{d_0, \ldots, d_{\overline{S}_i - 1}\}$ are sent in sequence from the first node of the route, that is $o_{r_i}(d_0) < o_{r_i}(d_1) < \ldots < o_{r_i}(d_{\overline{S}_i - 1})$.
For each route, the size of a slice is $\sum_{i=0}^{\overline{S}-1} |d_i|$. The amount of data sent by an RRH to its BBU is the same, regardless of the route. Hence, we assume that this size is the same for all route, and we denote it by τ.

Let us call $[t(S,u,r)]$ the set of tics used by a slice S on a route r at vertex u, that is $[t(S,u,r)] = \bigcup_{i=0}^{\overline{S}-1} \{t(d_i,u,r) + i \mid 0 \leq i < |d_i|\}$. Let r_1 and r_2 be two routes, on which two slices S_1 and S_2 are sent. We say that the two routes have a **collision** if they share an arc (u,v) and $[t(S_1,u,r_1)] \cap [t(S_2,u,r_2)] \neq \emptyset$.

In the context of cloud-RAN applications, we need to send a slice from an RRH u to a BBU v and then we must send the answer from v back to u. We say that a routed network $(G, \mathcal{R})$ is **symmetric** if the set of routes is partitioned into the sets F of **forward routes** and B of **backward routes**. There is a bijection ρ between F and B such that for any forward route $r \in F$ with first vertex u and last vertex v, the backward route $\rho(r) \in B$ has first vertex v and last vertex u. In all practical cases the routes r and $\rho(r)$ will be the same with the orientation of the arcs reversed, which corresponds to bidirectional links in *full-duplex* networks, but we do not need to enforce this property.

We now describe the process of the sending of a slice and of its answer. First, the datagrams of a slice S_r are sent at u, through the route $r \in A$, at time $\{o_r(d_0), \ldots, o_r(d_{\overline{S}_r - 1})\}$. Those datagrams are received by v, i.e., the last vertex of r at times $\{t(d_0,v,r), \ldots, t(d_{\overline{S}_r - 1},v,r)\}$. Once v has received all the

datagrams of a slice, the answer is computed and sent back in a slice S_{ρ_r}. Note that $\overline{S}_{\rho_r}$ may be not equal to $\overline{S}_r$, i.e. the answer is not necessarily under the same form as the initial slice. Thus the node v send S_{ρ_r} to u on the route $\rho(r)$. Note that, in the process we describe, we do not take into account the computation time a BBU needs to deal with one message. It can be encoded in the weight of the last arc leading to the BBU and thus we do not need to consider it explicitly in our model.

We define the **reception time** of a slice S_r on a node v by $r(S_r,v) = \max_{i\in\{0,\dots,\overline{S}_r-1\}} t(d_i,v,r)$. This is the time at which the last datagram of the slice has reach the vertex v.

The time between the arrival of the last datagram of a frame in the BBU and the time the answer is sent back is called the **waiting time** and is defined by $w_r = o_{\rho(r)}(d_0) - r(S_r,v)$.

The whole process time for a frame r is equal to $PT(r) = rt(S_{\rho_r},u) - o_r d_0$, where u is the RRH. In the process time, we count the time between the time the first tic of the frame is emitted and the time at which the last tic of the message comes back.Each route must respect a time limit that we call *deadline*. To represent these deadlines, we use a deadline function d, which maps to each route r an integer such that $PT(r)$ must be less than $d(r)$.

1.4 Periodic assignment for low latency

In our context of C-RAN, the process described above is periodic. The period P is and integer representing a number of tics. During each period, the same slcie is emited by each RRH/BBU on each route. A (P)-periodic assignment m is a choice of the offsets, and buffers of the datagrams of each slices such that each slice sent on the routed network satisfies its deadline, and does not collide with the others slices.

We consider the following decision problem.

Periodic Assignment for Low Latency (pall)

Input: A symmetric routed network $(G,\mathcal{R})$, the integers P and a deadline function d.

Question: does there exist a (P)-periodic assignment m of $(G,\mathcal{R})$ such that for all $r \in \mathcal{R}$, $PT(r) \leq d(r)$?

References