HOMA: Online In-Flight Service Provisioning with Dynamic Bipartite Matching

Amir Varasteh†, Saeed Akhoondian Amiri*, and Carmen Mas-Machuca†

†Chair of Communication Networks, TU Munich
Email: {amir.varasteh, cmas}@tum.de

*Department of Computer Science, University of Cologne
Email: amiri@cs.uni-koeln.de

Abstract—Airline companies are currently investigating means to improve in-flight services for passengers. Given emerging Air-to-Ground (A2G) communication technologies and the high desire of passengers for in-flight services, the servers providing in-flight services can be moved from the airplane to Data Centers (DCs) on the ground. In this scenario, network nodes (airplanes) demanding network services move over a ground core network. Therefore, the selection of DCs to connect to, as well as the underlying routing decisions are challenging. In particular, to keep a low-delay in-flight connection during the flight, airplanes connections can be reconfigured from a DC to another one, which comes at a delay cost.

This paper presents a formal model for the in-flight service provisioning problem, also as an Integer Linear Program (ILP). We show that the problem is NP-hard and hence propose an efficient online heuristic, HOMA, which addresses the above challenges in polynomial time. HOMA models the problem as a dynamic matching with special properties, and then efficiently solves it by a transformation into the shortest-path routing problem. Our simulation results indicate that HOMA can achieve near-optimal performance and outperform the baseline and state-of-the-art algorithms by up to 15% while reducing the runtime from hours to seconds.

Index Terms—Service Selection, Reconfiguration, Service Migration, Dynamic Service Allocation, Mobility-Aware, Mobile Users.

I. INTRODUCTION AND BACKGROUND

Today, with a significant increase in worldwide flight traffic volume, providing in-flight services such as entertainment to onboard passengers has become an important goal for airline companies [1]. Recently, there have been advances in providing high-capacity Air-to-Ground (A2G) communications to airplanes [2], [3]. In particular, Low Earth Orbit (LEO) satellites can relay the airplane traffic towards a gateway on the ground with a high link capacity [4]. Also, Direct-Air-to-Ground (DA2G) connections enable airplanes to communicate directly to the base stations located on the ground network [5], [6]. These developments have made new opportunities for airlines to deploy new online and interactive services, such as air-to-ground voice/video calls and gaming to improve passenger experience and grow the revenues [1]. To realize these services, airline companies can move the servers providing the in-flight services from the airplane to the distributed Data Centers (DCs) located on the ground. This can also reduce the weight of the airplane, hence the fuel consumption. Due to their real-time and interactive design, these types of services are usually demanding in terms of delay [7]. Thus, it is important to determine the DC to connect to as well as the underlying routing decisions to provide low-delay services to the airplane. Moreover, due to the mobility of airplanes over time, a DC may not remain the efficient one over time (i.e., the distance of the airplane to DC and hence the routing delay can increase during flight).

Therefore, to maintain a continuous low-delay service delivery to the airplanes, the airplane-to-DC assignments can be adapted over time, henceforth called reconfigurations1. While these reconfigurations can improve the airplane to DC routing delay, they come at a cost, for example, in terms of the

1In this paper, reconfiguring/reconfiguration and reassigning/reassignment terms are used interchangeably.
delay (e.g., airplane session’s data synchronization between two DCs). For clarification, an exemplary scenario with two flights is shown in Fig. 1.

Given a set of flights, their A2G connections, and a ground core network with capacitated DCs and links, it is of critical importance to choose the best DCs for airplanes to connect to using an efficient routing, as well as the necessary reconfiguration decisions during their flight period to keep the delay of in-flight service delivery minimized in a best-effort approach.

We see our work partially embedded in the Mobile Edge Computing (MEC) research area [8]–[10], with use-cases such as Internet-of-Vehicles (IoV) [11], [12]. Some works [10], [11] assuming that there is a single access point in the network and it is always available for the users to connect to, they focused on service placement on the available MEC servers. Further, by overlooking the end-to-end routing decisions, some works [8], [9] have focused on selecting the access point for the users. While in our case, in addition to selecting the A2G, we determine the routing from an airplane node to the DC with minimum delay. Also, in our scenario, the A2G availability can change over time, e.g., due to DA2G congestion or when an airplane is located out of DA2G communication range. This makes the joint A2G selection, and routing, considering the reconfiguration decisions more challenging. Moreover, some works [10], [11] ignored the capacity of the hosting resources, while we consider the DC capacity in our scenario. Further, in contrast to the existing works [11], we know the future positions of the users (airplanes), since the flight path is predetermined and static, at least in normal circumstances. This feature gives us opportunities to investigate novel approaches to tackle the mobility-aware resource management problems. Further, in the area of in-flight service provisioning, some works [6], [13], [14] have focused on offline algorithms for service-delivery cost optimization and A2G throughput improvement. However, an online algorithm with dynamic and low-delay service provisioning is missing. We believe an online solution would improve the network and resource allocations as it will adapt to late and/or unexpected flight changes with a low computational time.

In summary, the questions that we need to answer can be stated as follows:

1) At each timeslot, which airplane should be assigned to which DC? (Airplane-to-DC Connection)
2) When to reconfigure an airplane connection from a DC to another one? (Reconfiguration Decisions)
3) How to route the airplane traffic towards the selected DC? (Airplane-to-DC Routing)
4) How to jointly answer the above questions with minimum routing and reconfiguration cost (end-to-end delay) over the whole time horizon?

There are several challenges in designing this solution. Firstly, the designed solution needs to be efficient in terms of achieving minimum delay for the overall active flights in the network. Secondly, this algorithm needs to be fast, considering the incremental and dynamicity of the input (i.e., the arrival of unknown flight requests, and even emergency changes in the flight paths). Finally, the algorithm must meet different resource constraints such as network and DC capacities.

Thus, the contributions of this paper can be summarized as follows:

- First, we formulate the offline problem as an Integer Linear Programming (ILP) optimization model. Thereafter, by reducing the Generalized Assignment Problem (GAP) [17], we prove its NP-hardness. This shows that the ILP formulation is not scalable for large problem instances. However, the ILP formulation is still useful to compare the optimality of the proposed heuristics.

- To cope with the high complexity of ILP, we propose HOMA, an efficient online algorithm that addresses the in-flight service provisioning problem in polynomial time. We consider a flight-by-flight approach to solve the problem in online settings, where the flight requests are coming in sequence and with an unknown order. From the algorithmic viewpoint, our problem can be mapped to dynamic bipartite matching in graph theory [15], [16]. Specifically, in our scenario, a minimum-weight (routing delay) matching between airplanes and DCs needs to be maintained during the time, while the graph is changing dynamically over time due to the mobility of flights and hence, the availability of different A2G connections to them. Therefore, given a sequence of graphs as input, we formulate the problem into a dynamic bipartite matching problem with special properties. Having a sequence of bipartite matchings to be solved, we transform the matching and reconfiguration decisions in an auxiliary graph and use a shortest-path algorithm (e.g., Dijkstra) to solve the problem in a flight-by-flight fashion. We also show that HOMA finds the optimal solution in specific scenarios.

- Finally, we evaluate the performance of HOMA using extensive simulations in a European-based network with realistic settings. The simulation results show that the HOMA while significantly reducing the runtime, performs very closely to the optimal solution obtained from an ILP solver and outperforms the baseline and state-of-the-art algorithms.

**Outline.** We model and formulate the offline optimization problem in Sections II and III. In section IV, we introduce HOMA heuristic framework in detail. Then, we evaluate the performance of the proposed approach in Section V. Finally, Section VI discusses the related work, followed by the conclusions in Section VII.

**II. Problem Formulation**

In this section, we present the formulation of the problem, using the notations summarized in Table I. Let us start the mathematical formulation by defining dynamic graphs. Informally speaking, a dynamic graph is a graph that changes over time, either node and/or links. The formal definition is given by Definition 1.

**Definition 1.** A discrete and finite number of timeslots is defined as $T$. We define a dynamic graph as a sequence of $T$ static graphs $G_1 = (N_1, L_1), G_2 = (N_2, L_2), ..., G_T = (N_T, L_T)$.
The dynamic graph consists of four parts:

1) Flights: The set of flights is defined as $F = \{f_1, f_2, \ldots, f_{|F|}\}$. At each timeslot, each flight $f$ has a certain position over the earth (in terms of latitude and longitude), which is denoted by $p_{f,t}$. Also, each flight $f$ has a flying duration of $T_f$ timeslots. Each airplane location $p_{f,t}$ is a network node in $G_t$. Therefore, we define the set of airplane nodes as $N_f = \{p_{f,t} : f \in F, t \in T_f\}$ and $N_f \subset N_t$. Note that a feature of our problem is the prior knowledge of flight positions since the flight path is pre-determined and normally does not change. Moreover, the data rate of each flight (for all the passengers) is denoted by $b_{f,t}$ (e.g., in Mbps). These inflight services are considered to be interactive/real-time, such as air-to-ground voice/video calls, gaming, etc., with delay requirements of around 100 ms [7].

2) Ground Network: We define $N_C \subset N_t$ and $L_C \subset L_t$ as the set of nodes and links of the ground core network, respectively. These services are hosted by DCs located on the ground, defined as $N_{DC} \subset N_C$. A limited amount of resources are rented at each DC. Thus, each DC $j$ can process a maximum traffic volume of $C_j$. The duration of flight

3) DA2G Base Stations: We denote the set of DA2G base station nodes as $N_{DA2G} \subset N_t$, distributed/located on the ground. These nodes can directly communicate to the airplane flying over them and connect them to the ground network. Each DA2G base station has a communication range, usually between 150-250 km based on the technology that they use [17].

4) Satellite Connections: We use a single representative satellite node that relays the airplane traffic towards the satellite gateway located on the ground core network. We assume the satellite connection to be available to all the airplanes during the flight. Considering the dynamicity of the satellite constellations are out of the topic of this paper. Therefore, in graph $G_t$, we connect all the airplane nodes to the satellite node at all its timeslots. Fig. 2 shows an exemplary graph $G_t$ with three flights at a specific timeslot. Finally, we associate a constant propagation delay and a bandwidth capacity to each link $(u, v) \in L_t$, denoted by $D_{u,v}$ and $C_{u,v}$, respectively. We note that these values are different for each connection (link) type, i.e., core, satellite, and DA2G. The dynamicity of $G_t$ comes from the mobility of flights and their DA2G connections. As shown in Fig. 3, the availability of DA2G base stations around the airplane position can change due to a congestion model [13], hence the available links of the graph $G_t$ in a particular timeslot.

### III. INTEGER LINEAR PROGRAM FORMULATION

In this section, we present the offline optimization formulation as an Integer Linear Programming (ILP) model. Let us start with two delay functions:

1) Routing Delay: The routing delay (i.e., here delay) is the defined sum of the propagation delay of the path from the airplane to the selected DC in all timeslots, and is denoted by $T_{routing}$:

$$T_{routing}^G = \sum_{t \in T} \sum_{f \in F} \sum_{j \in N_{DC}} \sum_{(u,v) \in L_t} p_{f,t}^{u,v,j} D_{u,v}$$  

where $p_{f,t}^{u,v,j} \in \{0,1\}$ is a decision variable that is equal to 1 if the traffic of airplane node $p_{f,t}$ is routed towards DC $j$.

![Fig. 2: An instance of graph $G_t$ at a specific timeslot $t$. It is a realistic Europe-based core network with three DC nodes ($|N_{DC}| = 3$), and real DA2G base station locations. It shows three flights over Europe ($|N_f| = 3$) with the A2G connections: a representative satellite node and its gateway in Rome, and the connection to the distributed DA2G base stations in Europe.](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>Set of timeslots</td>
</tr>
<tr>
<td>$G_t = (N_t, L_t)$</td>
<td>Dynamic graph at timeslot $t \in T$; $N_t$ and $L_t$ are the set of nodes and links of the graph</td>
</tr>
<tr>
<td>$N_{DC}$</td>
<td>Set of data center nodes</td>
</tr>
<tr>
<td>$N_f$</td>
<td>Set of airplane nodes at time $t$</td>
</tr>
<tr>
<td>$C_j$</td>
<td>Capacity of DC $j$</td>
</tr>
<tr>
<td>$N_C$</td>
<td>Set of ground core network nodes</td>
</tr>
<tr>
<td>$L_C$</td>
<td>Set of ground core network links</td>
</tr>
<tr>
<td>$F$</td>
<td>Set of flights</td>
</tr>
<tr>
<td>$T_f$</td>
<td>The duration of flight $f \in F$</td>
</tr>
<tr>
<td>$p_{f,t}$</td>
<td>The position of flight $f$ at timeslot $t$</td>
</tr>
<tr>
<td>$b_{f,t}$</td>
<td>The traffic volume of flight $f$ at timeslot $t$</td>
</tr>
<tr>
<td>$D_{u,v}$</td>
<td>The delay of link $uv \in L$</td>
</tr>
<tr>
<td>$R_{i,j}$</td>
<td>Reconfiguration delay of assigning an airplane from DC node $i$ to $j$</td>
</tr>
<tr>
<td>$\Psi^+(i), \Psi^-(i)$</td>
<td>Outgoing and incoming nodes from and to node $i$</td>
</tr>
<tr>
<td>$D_{routing}$</td>
<td>Total routing delay</td>
</tr>
<tr>
<td>$D_{reconf}$</td>
<td>Total reconfiguration delay</td>
</tr>
<tr>
<td>$S$</td>
<td>Matching solution for a dynamic bipartite graph</td>
</tr>
<tr>
<td>$C(S)$</td>
<td>Total routing delay of solution $S$</td>
</tr>
<tr>
<td>$R(S)$</td>
<td>Total reconfiguration delay of solution $S$</td>
</tr>
<tr>
<td>$d(j)$</td>
<td>Degree of node $j$</td>
</tr>
</tbody>
</table>

### TABLE I: Notation definition.
We note that \( \delta \) is the satellite node or a DA2G node (if in range): airplane nodes towards a neighbor node in the graph, either the model. timeslot \( i \): DC node if the connection of airplane node \( r \) is used for both A2G options are accessible. between two DCs. This reassignment has a penalty in terms of \( p \) is connected only to the satellite, while at \( t = 2 \) (i.e., node \( p_{2,1} \)), it gets into a DA2G base station range, thus, both A2G options are accessible.

using the link \((u, v)\) at timeslot \( t \).

### 2) Reconfiguration Delay

Due to the mobility of airplanes, the reassignment of an airplane connection from a DC to another one requires state synchronization and data migration between two DCs. This reassignment has a penalty in terms of overhead traffic and effort, which is defined as the total reassignments delay for all the required reconfigurations \( D_{reconf} \):

\[
D_{reconf} = \sum_{t \in T} \sum_{f \in F} \sum_{(i,j) \in \mathcal{N}_{DC} \times \mathcal{N}_{DC}} r_{i,j}^{p_{f,t}} R_{i,j} \tag{2}
\]

where \( r_{i,j}^{p_{f,t}} \in \{0, 1\} \) is a decision variable that is equal to 1 if the connection of airplane node \( p_{f,t} \) is reconfigured from DC node \( i \) to \( j \) at timeslot \( t \). Also, \( R_{i,j} \) represents the time it takes for an airplane to change its connection from DC \( i \) to \( j \).

Let us now define the constraints that need to be met by the model.

**Assignment:** Assuming that flows are not splittable, each airplane must be assigned to one and only one DC at each timeslot \( t \):

\[
\sum_{j \in \mathcal{N}_{DC}} x_{j}^{p_{f,t}} = 1, \forall t \in T, \forall f \in F, \tag{3}
\]

where \( x_{j}^{p_{f,t}} \in \{0, 1\} \) is a binary decision variable which is equal to 1, if the airplane node \( p_{f,t} \) is assigned to DC \( j \) at timeslot \( t \).

**Traffic Generation:** The traffic is generated from the airplane nodes towards a neighbor node in the graph, either the satellite node or a DA2G node (if in range):

\[
\sum_{v \in \delta^+(p_{f,t})} \sum_{j \in \mathcal{N}_{DC}} l_{v,j}^{p_{f,t}} = 1, \forall t \in T, \forall f \in F. \tag{4}
\]

We note that \( \delta^+(i)/\delta^-(i) \) is a function that returns the outgoing/incoming links from/to node \( i \).

**Flow Conservation:** These constraints route the traffic flow through the network towards the assigned DC:

\[
\sum_{v \in \delta^+(u)} \sum_{j \in \mathcal{N}_{DC}} l_{u,v,j}^{p_{f,t}} - \sum_{v \in \delta^-(u)} \sum_{j \in \mathcal{N}_{DC}} l_{v,u,j}^{p_{f,t}} = \begin{cases} 0, & \forall t \in T, \forall f \in F, \forall u \in \mathcal{N}_{DC}, u \neq p_{f,t} \\ x_{u}^{p_{f,t}}, & \forall t \in T, \forall f \in F, \forall u \in \mathcal{N}_{DC} \end{cases} \tag{5}
\]

**Routing-Assignment Relation:** We need to ensure that the traffic flow of each airplane node at a particular timeslot is forwarded towards the assigned DC:

\[
l_{u,v,j}^{p_{f,t}} + u, v \in \mathcal{N}_{DC}, u \neq p_{f,t} \tag{6}
\]

In the above constraints, the traffic of the airplane node \( p_{f,t} \) can be routed towards DC \( j \), if and only if the airplane node is assigned to it at timeslot \( t \) (i.e., \( x_{j}^{p_{f,t}} = 1 \)).

**DC Capacity:** As mentioned before, the amount of airplane traffic that is being processed by each DC is limited:

\[
\sum_{j \in \mathcal{N}_{DC}} x_{j}^{p_{f,t}} b_{j,t} \leq C_{j}^{DC}, \forall t \in T, \forall j \in \mathcal{N}_{DC}. \tag{7}
\]

**Link Capacity:** The amount of the airplane traffic volume that can be delivered on a network link is limited to the link’s capacity. To express this, we define the following constraints:

\[
\sum_{j \in \mathcal{N}_{DC}} \sum_{i \in \mathcal{N}_{DC}} x_{j}^{p_{f,t}} b_{i,j} \leq C_{i,j}^{L}, \forall u, v \in \mathcal{N}_{DC}, \forall u \neq v \tag{8}
\]

**Reconfigurations:** The final set of constraints belongs to the reconfigurations that have to be performed:

\[
x_{j}^{p_{f,t}} = \sum_{i \in \mathcal{N}_{DC}} x_{j,i}^{p_{f,t}} - \sum_{i \in \mathcal{N}_{DC}} x_{i,j}^{p_{f,t}}, \forall t \in T \setminus \{0\}, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \tag{9}
\]

\[
x_{j}^{p_{f,t}} = \sum_{i \in \mathcal{N}_{DC}} x_{j,i}^{p_{f,t+1}} - \sum_{i \in \mathcal{N}_{DC}} x_{i,j}^{p_{f,t+1}}, \forall t \in (T \setminus \{t\}), \forall f \in F, \forall j \in \mathcal{N}_{DC}. \tag{10}
\]

Constraints (9) and (10) link \( x \) and \( r \) variables in a flow conservation way. If \( x_{j}^{p_{f,t}} = 0 \), the airplane node \( p_{f,t} \) is not assigned to DC \( j \) at timeslot \( t \), meaning no reconfigurations. \( x_{j}^{p_{f,t}} = 1 \) implies that a reconfiguration assigns the airplane node \( p_{f,t} \) to DC \( j \) at timeslot \( t \) and reassigns it at \( t + 1 \). This reassignment can be from the DC \( j \) to \( j \), which incurs zero reconfiguration delay; \( \forall i, j \in (J \times J) \), if \( i = j \), \( R_{i,j} = 0 \). We note that reconfiguration is not possible at the first timeslot.

Finally, the ILP formulation can be presented as follows. The objective function aims at minimizing the routing and reconfiguration delays over the whole time horizon.

\[
\text{minimize } \left(D_{routing}^{T} + D_{reconf}^{T}\right), \tag{11}
\]

s.t. Constraints (3) – (10),

vars: \( x_{j}^{p_{f,t}} \in \{0, 1\}, \forall t \in T, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \)

\( x_{j}^{p_{f,t}} \in \{0, 1\}, \forall t \in T, \forall f \in F, \forall j \in \mathcal{N}_{DC}, \)

\( x_{j}^{p_{f,t}} \in \{0, 1\}, \forall t \in T, \forall f \in F, \forall i, j \in \mathcal{N}_{DC}. \)
Theorem 1. The optimization model (11) is NP-hard.

Proof. Let us begin by defining the Generalized Assignment Problem (GAP). According to [18], GAP determines the minimum-cost assignment of n jobs to m agents, such that considering the capacity restrictions on the agents, each job is assigned to exactly one agent. Therefore, we are able to reduce from GAP to an instance of model (11) with \( T = 1 \):

\[
\text{Minimize } \sum_{j \in \mathcal{J}} \sum_{p \in \mathcal{P}_j} c_j^p x_j^p,
\]

s.t. \( \sum_{j \in \mathcal{J}} x_j^p = 1, \forall f \in \mathcal{F} \),

\( \sum_{j \in \mathcal{J}} x_j^p b_f \leq C_{j}^{\text{DC}}, \forall j \in \mathcal{N}_{\text{DC}}, \)

\( x_j^p \in \{0, 1\}, \forall f \in \mathcal{F}, \forall j \in \mathcal{N}_{\text{DC}}, \)

where \( c_j^p \) is the assignment cost of airplane node (job) \( p \) to DC (agent) \( j \) (e.g., the shortest-path delay value). This transformation is doable in polynomial time. If there exists an algorithm that solves the model (11), it solves the corresponding GAP as well. Given the NP-hardness of GAP [18], the model (11) must be NP-hard too. \( \square \)

IV. HOMA: The Heuristic Framework

In this section, we present HOMA, a deterministic approach that solves the problem mentioned in Section I in polynomial time. The proposed heuristic is formulated as a Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM) problem, as explained in Section IV.A. In this context, a bipartite matching should be maintained (considering the reconfiguration cost) for a sequence of graphs that may change over time dynamically. We solve the DC-MWBM in a flight-by-flight manner where a set of flight requests for the service provisioning during their flight time. This set of flight requests is assumed to arrive in an online manner. Also, the size of this set can be variable in different settings (later we assume it as 1 for the worst case). To solve the problem for a given flight, we transform the problem into a shortest-path routing problem, where using an auxiliary graph, the airplane-to-DC assignments, routing, and reconfiguration decisions are taken. These steps are explained in the following.

A. Dynamic Constrained Minimum-Weight Bipartite Matching (DC-MWBM)

Given the sequence of graphs \( G_t \) (see Fig. 3), we transform each \( G_t \) to a bipartite graph \( \mathcal{G}_t \). We define bipartite graph \( \mathcal{G}_t = (\mathcal{N}_t^F, \mathcal{N}_t^{\text{DC}}, \mathcal{L}_t) \). The set of links is defined as \( \mathcal{L}_t = \{(f, j, n): f \in \mathcal{N}_t^F, j \in \mathcal{N}_t^{\text{DC}}, n \in \delta^+(p_{f,t})\} \) which shows a matching from airplane node \( p_{f,t} \) to DC \( j \) through A2G node \( n \) (i.e., \( n \) can be satellite or DA2G connection). Each \( (f, j, n) \in \mathcal{L}_t \) is associated with a weight function \( w(f, j, n): L_t \rightarrow \mathbb{R}^+ \) which is defined as the routing delay between \( f \in \mathcal{N}_t^F \) and \( j \in \mathcal{N}_t^{\text{DC}} \) through the A2G node \( n \) (routing is performed in \( G_t \)). Since the position of airplanes changes per timeslot, the link weights from \( w \) function also change, i.e., due to the airplanes’ mobility, the routing delay between the airplane nodes and DCs changes. Considering that, the total number of links for each \( \mathcal{G}_t \) can be at most \( 2 \cdot |\mathcal{N}_t^F| \cdot |\mathcal{N}_t^{\text{DC}}| \). For clarification, Fig. 4 shows the bipartite graphs created for the scenario in Fig. 3. In this example, it is assumed that \( \forall f \in \mathcal{F}, |\mathcal{F}| = T = 3 \).

Having the sequence of bipartite graphs \( \mathcal{G}_t \), we define the DC-MWBM problem:

Definition 2. Given a sequence of bipartite graphs, find a sequence of per-graph matchings, such that the sum of total matching weights and their changes (i.e., reconfigurations) is minimized.

In this problem, we need to select a set of edges \( \mathcal{M}_t \subseteq \mathcal{L}_t \) such that each DC node \( j \) has a node degree at most equal to its capacity \( C_{j}^{\text{DC}} \). For simplification purposes, we consider the DC capacity as the maximum number of airplanes that can use the DC resources at the same time.

\[
d(j) \leq C_{j}^{\text{DC}}, \forall j \in \mathcal{N}_{\text{DC}},
\]

where \( d(j) \) indicates the node degree of node \( j \). Also, since we focus on the best-effort delay minimization, we relax the bandwidth constraints for now. However, we later show that in our solution, the bandwidth constraint can be addressed by simply removing the links with not enough available capacity.

Remark 1. Considering Hall’s theorem [19], \( \mathcal{G}_t \) can have a matching with cardinality \(|\mathcal{N}_t^F|\). Although \(|\mathcal{N}_t^F| >|\mathcal{N}_t^{\text{DC}}| \) in \( \mathcal{G}_t \); however, by making \( C_{j}^{\text{DC}} \) copies of nodes \( j \in \mathcal{N}_t^{\text{DC}} \) in \( \mathcal{G}_t \), we have \( \forall U \subseteq \mathcal{N}_t^F, |U| \leq \delta^+(U) \) (where \( \delta^+(U) \) is the set of DC nodes, i.e., copied nodes, that airplanes can connect to them). This condition is true by having the lower bound value for the DC capacity as \( \forall j \in \mathcal{N}_{\text{DC}}, C_{j}^{\text{DC}} = \left\lfloor \frac{|\mathcal{F}|}{|\mathcal{N}_{\text{DC}}|} \right\rfloor \) (and thus a feasible solution).

Hence, we ensure that all the airplane nodes in \( \mathcal{G}_t \) are matched (connected) to one DC node, i.e.,

\[
d(p_{f,t}) = 1, \forall p_{f,t} \in \mathcal{N}_t^F, t \in T.
\]

Having a sequence of dynamic bipartite graphs \( \mathcal{G}_t \), the solution of the offline DC-MWBM problem is a sequence of matchings \( \mathcal{S} = \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_T \), where \( \mathcal{M}_i \) is the matching of bipartite graph \( \mathcal{G}_i \). Similar to the objective function in ILP model (11), the cost of solution \( \mathcal{S} \) can be defined as follows:

1) Matching Cost (routing delay): The total matching cost of a solution \( \mathcal{S} \) is the sum of matchings (selected links in bipartite graphs) in all timeslots:

\[
C(\mathcal{S}) = \sum_{t \in T} \sum_{l \in \mathcal{M}_t} w(l),
\]

where \( l = (f, j, n) \in \mathcal{G}_t \). We remind that \( C(\mathcal{S}) \) is the sum for the routing delay of all the airplanes to the matched DC nodes in all timeslots.

2) Reconfiguration Cost: We define a per-link reconfiguration cost to model the cost of matching changes (i.e.,
reconfiguration delay). We denote the total reconfiguration cost of solution $S$ as $\mathcal{R}(S)$, defined as:

$$\mathcal{R}(S) = \sum_{t \in T} \sum_{(f,j) \in M_t} \sum_{j' \in M_{t-1}, (f = f', j \neq j')} R_{f,j,j'}.$$  
(16)

where $R_{f,j,j'}$ is the cost (delay) of reconfiguring an airplane connection from DC node $j'$ to $j$ in two consecutive timeslots. We note that reconfigurations are not possible at the first timeslot.

Having the cost functions in hand, our goal is to calculate $S$ for the sequence of $M_1, M_2, ..., M_T$, minimizing the total matching and reassignment costs:

$$\text{Minimize } \left( \mathcal{C}(S) + \mathcal{R}(S) \right).$$  
(17)

Similar to Theorem 1, the DC-MWBM can be proved to be NP-hard, which is omitted due to the lack of space.

### B. Solving DC-MWBM

In this subsection, we propose an approach to solve the DC-MWBM problem. At this stage, a request for service provisioning for a set of flights is given as an instance of DC-MWBM. To solve this matching problem, HOMA is designed to work in a flight-by-flight manner. In practice, when a flight is parked in an airport, before takeoff, the control center determines its flight path during the flight. HOMA uses this information (i.e., the airplane locations during the flight) to schedule the DC connections and reconfigurations for the whole flight period. Therefore, given a flight $f$ with duration $T_f$, we first build an auxiliary graph $\mathcal{G}_A$. We then introduce and solve a shortest-path routing problem in $\mathcal{G}_A$, which gives the solution of service assignment and reconfiguration problem for the given flight.

The main algorithm is aligned in Algorithm 1. The input of the algorithm is a single flight $f$ with duration $T_f$, for which, we find the airplane-to-DC assignment and the necessary reconfigurations at each timeslot. Also, to keep track of DC capacities, a dictionary with key-value pairs $(j, (t, C^j_{DC}))$, $j \in \mathcal{N}_{DC}$, $t \in T_f$ is calculated and given using the network status from previously served flights. In line 1, we initialize the sequence of bipartite graphs $\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T$ for the given flight $f$. In line 2, we create the auxiliary graph $\mathcal{G}_A$ as a multigraph. We create a dictionary in line 3 to keep the DCs with enough capacity at each timeslot (line 5). Thereafter, we start to build $\mathcal{G}_A$.

1) **Case 1, $t = 1$**: This is the case for the first timeslot of the flight $f$. In this case, we first add the airplane node $p_{f,1}$ to $\mathcal{G}_A$ as the first node (line 8). Then, we create a node for each available DC node (line 10), and connect the airplane node $p_{f,1}$ node at $t = 1$ to it (lines 11-15). These links represent an assignment decision, for instance link $(p_{f,1}, j_1)$ means airplane node $f$ is assigned to DC $j_1$ at timeslot $t = 1$.

As mentioned before, each airplane node can connect to a DC using one of the A2G connections (i.e., satellite or DA2G), which is determined in line 6. Therefore, for each available A2G (line 11), and add a link for each of them (line 15). For example, in Fig. 3a, it can be seen that at $t = 1$, flight $f = 2$ has only satellite connection. Therefore, in Fig. 5b
at \( t = 1 \), we connect the airplane node \( p_{2,1} \) to the DCs only using the satellite connection (dashed line). For each of these links, we define a weight, which will be used later to calculate the shortest path. These weights are defined as the delay of the shortest path between the airplane and DC nodes (in the original graph \( G_t \)) through the corresponding A2G connection. Note that, before calculating the shortest path, the links with lower than \( b_{f,t} \) available capacity are removed from the network. Therefore, in lines 12 and 13, we calculate the shortest-path delay from the airplane node \( p_{f,t} \) to the neighbor node \( n \) (i.e., either satellite or DA2G node), and from \( n \) to the DC node, respectively. We then sum up these two delay values (line 14) and finally create a link between the airplane and DC node with the total delay value as its weight (line 15).

2) \( 1 < t \leq T_f \): In this case, we continue building \( G_A \) by connecting DC nodes in two consecutive timeslots. We create a link between the DC nodes in timeslot \( t - 1 \) and \( t \), \( \forall 1 < t \leq T_f \) (lines 17-26). The link \((i_{t-1,j_t})\) which connected DCs \( i \) and \( j \) in \( G_A \), indicates that in two consecutive timeslots \( t - 1 \) and \( t \), the airplane nodes \( p_{f,t-1} \) and \( p_{f,t} \) are assigned to DC nodes \( i \) and \( j \), respectively. Compared to the first case, in addition to the assignment decisions, the possibility of reconfigurations is enabled here. When \( i \neq j \), the link represents a reconfiguration from DC \( i \) to \( j \) at timeslot \( t \). We loop through the previous DC nodes at \( t - 1 \) (line 17) and also at \( t \) (line 18). We create a node in \( G_A \) for all the available DC nodes \( j_t \) (line 19).

Similar to the first case, a maximum of two links can exist between nodes (per A2G connection type). We calculate the delay of the shortest-path between \( p_{f,t} \) and DC \( j_t \) through the neighbor node (i.e., A2G connections) \( n \) in \( G_t \). In case two DC nodes at \( t - 1 \) and \( t \) are not the same node in \( G_t \), it indicates a reconfiguration decision. Therefore, we add the reconfiguration cost between two DC nodes to the link weight (line 25). Finally, in line 26, we add a link to \( G_A \) from node \( j_{t-1} \) to \( j_t \) with the summed routing delay as the weight.

3) Case 3, dummy sink: This is the case where we add a dummy sink node \( s \) to \( G_A \) in line 27. Thereafter, we connect the DC nodes in timeslot \( t = T_f \) (last flight timeslot) to \( s \) with weight 0 (line 29).

Having the auxiliary graph \( G_A \) built, we can calculate the solution for the flight \( f \). To do this, we simply need to calculate the shortest path from \( p_{f,1} \) and the dummy sink node \( s \) nodes using conventional algorithms e.g., Dijkstra (line 30). The returned path gives us the assignment, routing, and reconfiguration decisions with minimum total delay. A comprehensive example is presented in Fig. 5. Finally, the DC capacities are updated in line 31 to be used for future flights.

Remark 2. Algorithm 1 can solve the problem for flights of any duration. In particular, we only need to keep track of DC capacities per timeslot. That is, the \( G_A \) can be formed for a flight with any duration. The number of nodes and links for \( G_A \) would depend on the flight duration.

Remark 3. For a given flight \( f \) with duration \( T_f \) timeslots, graph \( G_A \) can have \( O(T_f \cdot |N_{DC}|) \) nodes and \( O(T_f \cdot |N_{DC}|^2) \) links.

Lemma 2. For any given flight \( f \) with duration \( T_f \), HOMA is complete\(^2\) and optimal.

Proof. For a given flight \( f \), we build the auxiliary graph \( G_A \) with the available DC nodes. Also, we add all the possible routing options (DA2G and/or satellite) with the weight defined as the shortest-path delay between the airplane and DC. Also, we consider all possible assignments and reconfiguration decisions in \( G_A \). Therefore, since Dijkstra is complete and

\(^2\)An algorithm is complete if it always finds a solution if one exists. The completeness does not imply optimality.
For any given set of flights $F$, if $\forall j \in N_{DC}, C_{j}^{DC} \geq |F|$, HOMA is complete and optimal.

Proof. Flights in $F$ depend on each other through the DC capacity. If $\forall j \in N_{DC}, C_{j}^{DC} \geq |F|$, it simply means at each timeslot, a single DC node can host all the flights. Thus, the DC capacity constraints can be relaxed in this case. Therefore, according to the Lemma 2, HOMA is complete and optimal for all the set of flight $F$.

Theorem 3. For any given flight $f$, the complexity of HOMA for finding the optimal solution is $O(T_f \cdot |N_i|^2)$.

Proof. We calculate the runtime based on the number of times that the Dijkstra algorithm needs to be performed to find the solution for flight $f$. The Dijkstra algorithm needs to be executed for each link in the auxiliary graph $G_A$, except the links for the dummy destination node $s$. Therefore, considering Remark 3, we need $2 \cdot (|T_f| - 1) \cdot |N_{DC}|^2 + |N_{DC}|$ links (i.e., Dijkstra runs) to build the $G_A$. Each Dijkstra run is executed for $G_i = (N_i, L_i)$, with runtime $O(|N_i|^2)$. Since $|N_{DC}| \ll |N_i|$, the overall algorithm runtime for any flight $f$ can be expressed as $O(T_f \cdot |N_i|^2)$. We note that in use-cases where the duration of the timeslots is very small (close to 0), the algorithm complexity is pseudo-polynomial (based on $T_f$).

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of HOMA in various scenarios and settings. We first introduce the algorithms that we compare with the optimal offline (ILP) solution. Thereafter, we explain the simulation setup, input parameters, and scenarios. Finally, we present the results of the performance evaluation of HOMA against the offline optimal solution (section III) and the two baseline algorithms.

A. Algorithms to Compare

For comparison purposes, we re-implemented the algorithm presented in the state-of-the-art work [11]. Further, we develop two variations of HOMA (Minimize Delay - MD, and Maintain Connection - MC) as the baseline algorithms. Below, we explain these approaches in short.

1) Threshold-Based Reconfiguration (TBR): The first algorithm to compare with HOMA is taken from the state-of-the-art work [11]. In [11], the authors propose a resource management framework for service placement and migration for the automotive use case, within the Mobile Edge Computing (MEC) scenario. Similar to us, they have focused on providing low-delay services to the mobile users (cars in their case), while determining the appropriate MEC server to host the services, and the necessary reconfigurations according to the mobility of the cars. In their approach, they set a threshold for which, if the current routing delay value is lower than a threshold value, the service is migrated to another MEC server. This MEC server (destination of the migration) is chosen as the farthest one, to lower the number of future migrations. For comparison purposes, we have chosen two threshold values, 20 and 40 ms. We show these two algorithm versions as TBR-20 and TBR-40, respectively.

However, in contrast to us, the approach presented in [11] lacks considering a few constraints. Firstly, they do not consider the DC and link capacities into the account. Secondly, they ignore the service migration (reconfiguration) penalty in their algorithm. Nevertheless, for a fair comparison, we have added these considerations into the implementation of their algorithm.

2) Minimize Delay (MD): The next algorithm is a baseline that focuses on minimizing the routing delay between the airplane nodes and the assigned DC. MD reconfigures the airplane connections whenever it leads to a lower routing delay. At $t = 1$, it chooses the DC with the lowest routing delay to connect the airplane to it. For $t > 1$, it reconfigures the airplane connection to a DC which keeps the routing delay at that specific timeslot minimized. Regarding the implementation, it is enough to set $\forall i,j \in N_{DC}, R_{i,j} = 0$, i.e., reconfiguration becomes a decision with no cost. Thus, in the procedure of building $G_A$, in the case 2 where $t > 1$, we update the weight of link $(i,j), i,j \in N_{DC}$ with zero reconfiguration cost, i.e., changing line 25 to $\text{sum}_\text{delay} += 0$ in Algorithm 1.

3) Maintain Connection (MC): As it appears from its name, this algorithm always maintains its connection to the first selected DC. The goal of developing this algorithm is to highlight the importance of providing flexibility in the network (i.e., reconfigurations). At the first timeslot of the flight, MC connects the airplane to the DC with minimum routing delay and keeps using it for the rest of the flight duration. The implementation is done with a small tweak in Algorithm 1, in Case 2, where $t > 1$. In particular, while building the $G_A$, in two consecutive timeslots $t-1$ and $t$, MC adds links only between DC nodes $i_{t-1}$ and $j_t$, if $i = j$ where $i,j \in N_{DC}$. In this way, the reconfiguration of airplane connections would not be possible for the Algorithm (since there are no links between different DCs in $G_A$).

B. Simulation Setup

In this subsection, we explain the simulation setup and input parameters used in the simulations for evaluating the performance of the proposed algorithm.

The set of flight requests (i.e., $F$) was exported from FlightRadar24 live air traffic for 24 hours on 9.11.2017. We extracted the position of flights at each timeslot and use it as $p_{t,i}$ parameter. In our experiments, we select a set of flights $F$, such that $|F| = \{20, 50, 100, 300, 500\}$. For simplification, we assume that all the airplanes are transmitting the same amount of traffic $b_{f,t}$ during all the timeslots. As the ILP takes several hours to solve a problem instance with 100 flights, the number of timeslots has been limited to eight, so we can have problem instances with a high number of flights. Thus, we choose $F$ with two different flight duration, long and short flights, with four and eight timeslots, respectively. For simplicity, the duration of each timeslot is considered 30
minutes (this point is discussed in section V-D.D). Indeed, the number of timeslots depends on the airplane speed and the range of the DA2G and satellite coverage, which can lead to an even different number of timeslots per flight. However, to simplify the evaluation scenario, we consider the same number of timeslots and duration for all the flights.

The simulation settings are based on a realistic European-based flight space (See Fig. 2). We use the European Cost266 topology [22] for the ground core network and set the values of $N_C$ and $L_C$ accordingly. The delay of links $(D_{u,v}, u, v \in N_C)$ in $L_C$ is determined according to the length of the optical fiber transmissions. Regarding the DC nodes, we choose two sets of locations, distributed over Europe [23]: a small set $N_{DC}=$ {Hamburg, Madrid, Budapest}, and a larger set $N_{DC}=$ {Strasbourg, Stockholm, Madrid, Athens, Glasgow, Krakow}. We set the value of DC capacity $C^{DC}$ by considering three cases:

1) **low**: $\forall j \in N_{DC}, C^{j, DC} = \left\lfloor \frac{|F|}{|N_{DC}|} \right\rfloor$ (note that lower DC capacity makes the problem infeasible, see Remark 1).
2) **high**: it sets the DC capacity equal to the total number of flights in the network $\forall j \in N_{DC}, C^{j, DC} = |F|$.
3) **medium (med)**: the average of low and high values.

Since the focus of this work is to provide (best-effort) low-delay in-flight services, we consider $\forall f \in F, t \in T_f, b_{f,t} = 1$ with a relaxation on the bandwidth constraints for the A2G and ground network links. There are 295 DA2G base stations (already deployed in Europe), and their location (in terms of latitude and longitude) is taken from [24]. Also, the propagation delay from airplane to the DA2G base station is set to 10 ms [17] (i.e., $D_{u,v}, u \in N^{f}_I, v \in N_C$ through the satellite node). This value is calculated according to the LEO delay calculations provided in [27]. Also, the satellite gateway node is considered to be located in Rome, Italy [28]. An instance of the built European-based scenario can be seen in Fig. 2 (for a single timeslot).

Regarding the reconfiguration overhead, we use the distance between source and destination DCs:

$$R_{i,j} = \begin{cases} M \ast \text{dijkstra\_path\_length}(i,j), & i,j \in N_{DC} \ (i \neq j), \\ 0, & i,j \in N_{DC} \ (i = j), \end{cases}$$

$M$ is a random number between 0 and 2. We note that the modeling of the service migration is out of the scope of this paper. However, different $R_{i,j}$ models can be plugged into the HOMA, depending on the specific application.

To design different experiments, we use the input parameters and their values in Table II where in each experiment, three parameters are fixed and the fourth one is varied. We implement and solve the ILP using Gurobi [29]. Also, HOMA, TBR, MC, and MD approaches are implemented in Python.

We perform the evaluations for 30 random sets of scenarios on a machine equipped with Intel Core i7-6700 CPU3.40 GHz, 16 GB RAM, and running Arch Linux with kernel 5.5.11-arch1-1.

### C. Simulation Results

In this subsection, we present the simulation results for different parameters and experiments.

1) **Number of Flights**: We first consider the default values for flight duration ($T_f$), number of DCs ($|N_{DC}|$), and DC capacity ($C^{DC}$) according to Table II. The effect of number of flights on total delay (i.e., the objective function value), routing delay, and number of reconfigurations can be seen in Fig. 6a-6c. In Fig. 6a, it can be seen that HOMA achieves a near-optimal objective function value with around 1% of distance from the optimal. Also, HOMA outperforms the baseline and state-of-the-art algorithms for up to 15% on average for a high number of flights. The MD algorithm achieves a lower routing delay compared to HOMA (see Fig. 6b) since it greedily reconfigures the airplane connection to the closest DC to minimize the routing delay. However, it requires a significant number of reconfigurations to be able to minimize the airplane to DC delay (See Fig. 6c). Further, MC leads to a larger routing delay compared to HOMA, since it maintains the connection of the airplanes to the first selected DC for the whole flight period and does not reconfigure the connection. Therefore, during the flight, when the airplane gets far from the DC, the routing delay also increases. Also, it can be seen that the TBR-20 and TBR-40 lead to a higher objective value function. The reason is in TBR, determining the threshold value for triggering the reconfiguration is challenging, especially when consider the LEO constellation for the evaluation scenario. For simplification purposes, we apply an abstraction model, where the group of LEO satellites occupying Europe is represented as a single satellite node. However, to avoid unrealistic assumptions, we set the latency of the satellite link to the worst-case achievable latency 50 ms (i.e., $D_{u,v}, u \in N^{f}_I, v \in N_C$ through the satellite node). This value is calculated according to the LEO delay calculations provided in [27]. Also, the satellite gateway node is considered to be located in Rome, Italy [28]. An instance of the built European-based scenario can be seen in Fig. 2 (for a single timeslot).

Regarding the reconfiguration overhead, we use the distance between source and destination DCs:

$$R_{i,j} = \begin{cases} M \ast \text{dijkstra\_path\_length}(i,j), & i,j \in N_{DC} \ (i \neq j), \\ 0, & i,j \in N_{DC} \ (i = j), \end{cases}$$

$M$ is a random number between 0 and 2. We note that the modeling of the service migration is out of the scope of this paper. However, different $R_{i,j}$ models can be plugged into the HOMA, depending on the specific application.

To design different experiments, we use the input parameters and their values in Table II where in each experiment, three parameters are fixed and the fourth one is varied. We implement and solve the ILP using Gurobi [29]. Also, HOMA, TBR, MC, and MD approaches are implemented in Python.

We perform the evaluations for 30 random sets of scenarios on a machine equipped with Intel Core i7-6700 CPU3.40 GHz, 16 GB RAM, and running Arch Linux with kernel 5.5.11-arch1-1.

### C. Simulation Results

In this subsection, we present the simulation results for different parameters and experiments.

1) **Number of Flights**: We first consider the default values for flight duration ($T_f$), number of DCs ($|N_{DC}|$), and DC capacity ($C^{DC}$) according to Table II. The effect of number of flights on total delay (i.e., the objective function value), routing delay, and number of reconfigurations can be seen in Fig. 6a-6c. In Fig. 6a, it can be seen that HOMA achieves a near-optimal objective function value with around 1% of distance from the optimal. Also, HOMA outperforms the baseline and state-of-the-art algorithms for up to 15% on average for a high number of flights. The MD algorithm achieves a lower routing delay compared to HOMA (see Fig. 6b) since it greedily reconfigures the airplane connection to the closest DC to minimize the routing delay. However, it requires a significant number of reconfigurations to be able to minimize the airplane to DC delay (See Fig. 6c). Further, MC leads to a larger routing delay compared to HOMA, since it maintains the connection of the airplanes to the first selected DC for the whole flight period and does not reconfigure the connection. Therefore, during the flight, when the airplane gets far from the DC, the routing delay also increases. Also, it can be seen that the TBR-20 and TBR-40 lead to a higher objective value function. The reason is in TBR, determining the threshold value for triggering the reconfiguration is challenging, especially when consider the LEO constellation for the evaluation scenario. For simplification purposes, we apply an abstraction model, where the group of LEO satellites occupying Europe is represented as a single satellite node. However, to avoid unrealistic assumptions, we set the latency of the satellite link to the worst-case achievable latency 50 ms (i.e., $D_{u,v}, u \in N^{f}_I, v \in N_C$ through the satellite node). This value is calculated according to the LEO delay calculations provided in [27]. Also, the satellite gateway node is considered to be located in Rome, Italy [28]. An instance of the built European-based scenario can be seen in Fig. 2 (for a single timeslot).

According to [26], the number of LEO satellites existing over Europe is rather low (5 out of the 72 LEO satellites in the Iridium constellation). Compared to the Geostationary Orbit (GEO) satellites, LEO causes lower delay, which is more suitable for our use case. Therefore, in this paper, we only consider the LEO constellation for the evaluation scenario. For simplification purposes, we apply an abstraction model, where the group of LEO satellites occupying Europe is represented as a single satellite node. However, to avoid unrealistic assumptions, we set the latency of the satellite link to the worst-case achievable latency 50 ms (i.e., $D_{u,v}, u \in N^{f}_I, v \in N_C$ through the satellite node). This value is calculated according to the LEO delay calculations provided in [27]. Also, the satellite gateway node is considered to be located in Rome, Italy [28]. An instance of the built European-based scenario can be seen in Fig. 2 (for a single timeslot).

### C. Simulation Results

In this subsection, we present the simulation results for different parameters and experiments.

1) **Number of Flights**: We first consider the default values for flight duration ($T_f$), number of DCs ($|N_{DC}|$), and DC capacity ($C^{DC}$) according to Table II. The effect of number of flights on total delay (i.e., the objective function value), routing delay, and number of reconfigurations can be seen in Fig. 6a-6c. In Fig. 6a, it can be seen that HOMA achieves a near-optimal objective function value with around 1% of distance from the optimal. Also, HOMA outperforms the baseline and state-of-the-art algorithms for up to 15% on average for a high number of flights. The MD algorithm achieves a lower routing delay compared to HOMA (see Fig. 6b) since it greedily reconfigures the airplane connection to the closest DC to minimize the routing delay. However, it requires a significant number of reconfigurations to be able to minimize the airplane to DC delay (See Fig. 6c). Further, MC leads to a larger routing delay compared to HOMA, since it maintains the connection of the airplanes to the first selected DC for the whole flight period and does not reconfigure the connection. Therefore, during the flight, when the airplane gets far from the DC, the routing delay also increases. Also, it can be seen that the TBR-20 and TBR-40 lead to a higher objective value function. The reason is in TBR, determining the threshold value for triggering the reconfiguration is challenging, especially when

### TABLE II: A summary of important input parameters and their values used for the performance evaluation. The default value is in bold text.

<table>
<thead>
<tr>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of flights ($</td>
<td>F</td>
</tr>
<tr>
<td>Flight duration ($T_f, \forall f \in F$)</td>
<td>4, 8</td>
</tr>
<tr>
<td>Number of DCs ($</td>
<td>N_{DC}</td>
</tr>
<tr>
<td>DC capacity ($C^{DC}, \forall j \in N_{DC}$)</td>
<td>low, medium, high</td>
</tr>
</tbody>
</table>
the goal is to minimize the overall delay. Lower threshold values (i.e., TBR-20) can lead to higher total delays and more reconfigurations are triggered for TBR (see Fig. 6c). According to its design, TBR reconfigures the service to the farthest DC, which increases the routing delay. In addition, since TBR does not consider the reconfiguration cost into the account, the balance between the routing delay and reconfiguration delay is missed, hence, leading to sub-optimal decisions. Fig. 6c shows the number of per-flight reconfigurations for the flight duration, in which, the proposed HOMA approach can make near-optimal reconfiguration decisions for the flights. Also, as expected, MC does not perform any reconfigurations.

All in all, HOMA can accurately balance the airplane to DC routing delay and reconfigurations, demonstrating a near-optimal behavior, while outperforming the baseline and state-of-the-art algorithms. The algorithms exhibit similar behavior with short flights. However, we omit these results from the paper due to space limitations. We note that we can derive the average round-trip delay per flight per timeslot from Fig. 6a, which is around 68 ms in the worst case for 500 flights for the proposed HOMA approach (requirements being around 100 ms [7]), which can support a wide range of interactive services such as air-to-ground voice calls, video conferencing, and gaming [7].

2) Number of DCs: Fig. 7a-7c show the evaluation results of the algorithms for a varied number of DCs, while keeping the default values for the other parameters. The goal here is to compare the behavior of different approaches when the number of available DCs changes. This can assist operators in more accurate capacity planning. Regarding the total delay, Fig. 7a indicates that generally, a lower number of DCs leads to a higher delay value. That is because, with a higher number of DCs, airplane nodes get in a closer distance to a DC node on average during their flight. This can be seen also in Fig. 7b, where the routing delay is generally lower in the case with six DC nodes. However, the TBR algorithm does not follow this trend, with each reconfiguration, the routing delay increases (since the reconfiguration is done towards the farthest DC).

Also, it can be seen that the difference in total delay achieved by the algorithms, is generally lower for three DCs compared to six. This is because the role of DC capacity (which is the main cause of sub-optimality) is more critical with a higher number of DCs (i.e., there is more room to be sub-optimal).

However, regarding the number of reconfigurations, Fig. 7c shows that the case with six DCs has around 20% more reconfigurations compared to the case with three, since it brings more flexibility and more opportunities for reconfiguration, especially for the TBR algorithm. Of course, this value gets lower with services with higher reconfiguration delay. However, still, the case with three DC makes more sense, since the proposed approach, HOMA, leads to low-delay solutions, while the number of reconfigurations is kept low. The results show that using HOMA can reduce the operators’ costs, by renting/building fewer DCs, while providing an good solution quality.
3) DC Capacity: In this study, we use the default values for the number of flights, flight duration, and the number of DC nodes, and vary the DC capacity as per Table II. The goal of this study is to find out what is the impact of different DC capacities on the performance of different algorithms. Fig. 8a shows the comparison of the achieved total delay by the algorithms for low, medium, and high DC capacities. Since our approach is online (flight-by-flight), the DC capacity can play a big role in having the available resources in a good DC location when needed. It can be seen that generally, the behavior of the algorithms is similar for the cases with different DC capacities. However, the total delay decreases with higher DC capacity. Also, the same trend is there for the routing delay as in Fig. 8b. The reason behind this is there are more DCs available to serve the airplanes (i.e., the average distance between airplanes and DCs is decreased). Moreover, as proved in Theorem 3, it is evident that HOMA is able to take optimal decisions for DC selection and reconfiguration in case of high DC capacity. Finally, Fig. 8c shows that the DC capacity does not change the number of reconfigurations much for the optimal and HOMA, i.e., they stay around one reconfiguration per flight for all the DC capacities.

These evaluations show that it is not necessary to rent plenty of resources at each DC. Medium or even low amount of resources could provide a good solution. This indicates that in contrast to other algorithms, HOMA can lead to cost savings for the operators.

4) Runtime: The runtime comparison of the optimal offline solution (i.e., ILP), HOMA, MD, MC, and TBR algorithms is presented in Table III. Firstly, this table shows that the ILP formulation is significantly slower than the other heuristics. Also, it can be seen that HOMA can solve the problem for 500 flights in around six seconds per flight. Also, MC has the lowest runtime, since it does not consider the reconfiguration decisions, thus the number of Dijkstra runs is considerably lower, compared to the HOMA, and MD approaches. Finally, the TBR algorithms are the fastest ones, since they are less sophisticated in nature.

D. Remarks

Let us summarize some remarks related to the design of HOMA. Firstly, we choose to adopt the discrete-time model mainly due to its simplicity. Furthermore, this model can be arbitrarily close to the continuous-time model by making the timeslot duration very small. However, it is not the case for our scenario. Considering the airplane speed and the A2G coverage range, the decision-making can be done in longer periods (on a scale of a few minutes).

Secondly, we consider the flight paths to be a priori knowledge, once planes are ready to take off. In practice, the paths can be changed in some emergencies (e.g., bad weather conditions). In this case, HOMA can be rerun to adapt the previously-made decisions using the updated network status (e.g., network link availability/congestion status) according to the emergency.

Third, in this work, the LEO satellite constellation has been modeled with a single representative node with the worst-case delay value. The integration of the dynamic satellite network considering the cost/delay of the satellite handover could be added to our approach as follows: the handover cost between two consecutive timeslots could be added to the cost of satellite connection links (i.e., network link availability/congestion status) according to the emergency.

Finally, we point out that HOMA does not guarantee the service-specific delay requirements but follows a best-effort approach toward delivering low-delay in-flight services to the airplane. However, the routing decision-making part of HOMA can be enhanced with deterministic networking theories such as Network Calculus [30], to provide a delay guarantee for the critical traffic flows (e.g., flight status).

VI. RELATED WORK

The related work can be discussed in different categories. A closely-related domain to our work is space-air-ground integrated (terrestrial) networks. [31], [32]. In this area, there
are studies that have focused on resource allocation [13], [14], [25], [33]–[35], satellite-based vehicular networks [36], reliability [37], and energy-efficiency [38]. For example, Varasteh et. al. [13], [25] has presented an offline mobility-aware optimization formulation for the joint service placement, routing, and migration for flying airplanes. Further, in [35], they have extended their work to use reinforcement learning for decision making. Moreover, authors in [14], [34] have focused on the onboard A2G communication analysis. In particular, in [14], the authors have performed a throughput analysis of DA2G communication during its flight. Also, [34] has presented a QoS-aware approach to satisfy the passengers' traffic flows using different A2G alternatives during a flight.

On the other hand, some works have focused on the resource allocation and management of MEC environments [39]–[43]. In [39], authors have proposed a graph-based algorithm to determine the number of servers, their size, and the operation area. Considering a maximum resource capacity for the MEC servers, their goal is to serve the maximum number of users at the edge area. Similar to them, the authors in [40] tackled the problem of edge server placement. They have proposed a multi-objective formulation for balancing the workloads on the MEC servers and reducing the delay between the clients (industrial control centers in their case) and MEC servers.

A closer area to our work is where the works have considered the resource allocation reconfiguration problem with mobile users [8]–[10], [44]–[52]. Authors in [46], have proposed a mobility-aware service placement approach, considering the mobility of users and location, to minimize the service access delay and deployment costs. Further, some works [49], [51] have studied the dynamic placement of network functions on MEC servers according to the future handover probability of users. Specifically, [49] proposes two service migration solution for MEC-based applications: reactive, and proactive. There are other works with a focus on proactive service provisioning using a look-ahead window and pre-allocation techniques [43], [47]. Also, authors in [10] have proposed a Virtual Machine (VM) migration method, called Follow-Me Cloud, to deal with user mobility. In more detail, the aim of them is to always connect the mobile users to the optimal gateway, while the service instance (i.e., VM) follows the users, using the migration technologies. They have proposed an approach based on the Markov decision process to determine the appropriate migration decisions. However, they do not consider the network and servers capacity limits.

In the IoV area, there are similar works [11], [12], [53], [54] that provide service continuity to users (vehicles), considering their mobility. For instance, Zhang et. al in [53] have focused on the placement of a constant number of edge servers to improve the QoS (i.e., reducing the average waiting time of services in the IoV), and load balancing. Also, the authors in [11] present a 5G-enabled framework that migrates the services based on the mobility of the vehicles along the road. They keep monitoring the delay between the vehicle and the MEC server. In case the delay goes higher than a threshold, they migrate the service to another MEC in the direction of the vehicle movement. However, these works overlook the end-to-end routing, reconfiguration overheads, and resource capacities. Also, some works [10], [51] assume that the access points are always available to the users to connect to. Also, some of them overlooked the end-to-end routing decisions [8], [9], [11], [50], DC capacity [8], [9], [11].

Finally, from an algorithmic viewpoint, this work is related to the matching problem in graph theory. The matching problem has a long history of research, going back to decades ago [55], [56]. Due to the enormous applications of matching in practice [57], there is a huge body of research for online (bipartite) static matching and its variants [58]–[65], [66]. However, due to the dynamic nature of many matching problems in practice (e.g., considering the temporal/dynamic graphs), recently, dynamic matching has been receiving attentions [15], [16], [67]–[69]. Despite the similarity in names, our problem in this paper is different from others. In our problem, there is a sequence of graphs that change dynamically due to airplanes’ mobility. Therefore, the goal is to maintain a minimum-weight bipartite matching, considering the mobility of airplanes.

VII. CONCLUSIONS

This paper initiated the study of an online service provisioning problem in the context of emerging air-to-ground communication technologies. We presented a formal model together with an efficient algorithm, HOMA to decide about airplane-to-DC connections, routing, and possible reconfiguration decisions over the flight period. HOMA showed a good performance in realistic scenarios, achieving a near-optimal objective function value, while significantly reducing the runtime. We see our work as a first step and believe that it opens several interesting avenues for future research. In particular, it would be interesting to explore the achievable competitive ratio in deterministic and randomized online settings and explore how to optimize and re-optimize schedules in scenarios that feature some uncertainty and in which a bounded number of unexpected events may occur.

We note that a similar approach can be applied to other areas such as Autonomous Driving where a vehicle needs to travel from point A to B in a city. Considering the positions of the vehicle at different times, the required services can be scheduled on the MEC servers available in the city.

ACKNOWLEDGMENT

The authors would like to thank Nemanja Deric, Stefan Schmid, and Mojgan Madadikhaljan for their support and constructive feedback. This work is funded by Germany Federal Ministry of Education and Research under project AI-NET ANTILLAS (project ID #16KIS1318).

REFERENCES
