Age-minimal Multicast by Graph Attention Reinforcement Learning

Yanning Zhang
Zhejiang University

Guocheng Liao
Sun Yat-Sen University

Shengbin Cao
University of Macau

Ning Yang
Chinese Academy of Sciences

Meng Zhang
Zhejiang University

***Abstract*—*Age of Information (AoI)* is an emerging metric used to assess the timeliness of information, gaining research interest in real-time multicast applications such as video streaming and metaverse platforms. In this paper, we consider a dynamic multicast network with energy constraints, where our objective is to minimize the expected time-average AoI through energy-constrained multicast routing and scheduling. The inherent complexity of the problem, given the NP-hardness and intertwined scheduling and routing decisions, makes existing approaches inapplicable. To address these challenges, we decompose the original problem into two subtasks, each amenable to reinforcement learning (RL) methods. Subsequently, we propose an innovative framework based on graph attention networks (GATs) to effectively capture graph information with superior generalization capabilities. To validate our framework, we conduct experiments on three datasets including a real-world dataset called AS-733, and show that our proposed scheme reduces the average weighted AoI by 62.9% and reduces the energy consumption by at most 72.5% compared to baselines.**

I. Introduction

A. Background and Motivations

Real-time multicast applications, such as video streaming [1] and intelligent transportation systems [2], have experienced significant growth in recent times. These applications require timely updates to ensure accurate and up-to-date information availability for critical tasks like decision-making and system control. While delay is a commonly used metric in traditional networks [3], it is now recognized that ensuring timely updates is distinct from simply minimizing delay [4]. Consequently, there is a need for a metric that captures the timeliness aspect of information dissemination. The concept of *Age of Information (AoI)* has emerged as a promising metric in various domains, including learning and network protocols [5]. AoI quantifies the freshness of information possessed by a monitor about a specific entity or process, which has been identified as a suitable metric for evaluating the performance of multicast networks [6], making it particularly relevant for real-time multicast applications. It has been shown that AoI is the most important metric for evaluating the Quality of Service (QoS) in some scenarios [7].

Multicast, a vital communication paradigm in networks, facilitates the efficient dissemination of information from a source to multiple destinations. Recent research has extensively investigated the applications of multicast in various scenarios. For instance, optimizing the multicast Quality of Experience (QoE) is crucial for enhancing video streaming sessions [8]. One of the primary problems lies in the routing process, which entails determining the optimal paths from the source to destinations. The routing problem falls within the domain of Combinatorial Optimization (CO) problems, such as the Steiner tree problem [9], known for their NP-hardness, rendering them computationally demanding to solve in large-scale networks.

Due to the inherently distributed feature of network systems, only local information is available for a centralized controller, making it challenging to optimize the routing process. Fortunately, recent advances in Software Defined Networking (SDN) enabled the intelligent control of network devices [10]. SDN is a network architecture that separates the control plane from the data plane, where the control plane is centralized and is responsible for managing network resources and programming the network dynamically. The centralized controller has a global view of the network by monitoring and collecting the real-time network state (e.g., packets) and configurations. The above features ensure that the solutions given by intelligent algorithms (e.g., AI-based algorithms) can be implemented in real-world scenarios.

Furthermore, real-world networks often operate under energy constraints, where the overall energy consumption of the network is limited. This introduces additional complexity to the problem, as there exists a trade-off between energy consumption and AoI [11]. Consequently, certain existing algorithms (e.g., [12]) become inapplicable. Some studies have proposed scheduling algorithms to optimize AoI [6]. However, they tend to overlook the routing problem, which is also crucial for AoIs. Hence, there is a clear demand for a novel multicast scheme that offers a comprehensive solution to optimize AoI in energy-constrained multicast networks.

In this paper, we aim to answer the following main question: *How should one design multicast scheduling and routing algorithms to make the optimal tradeoffs between AoI and energy consumption?*

B. Key Challenges

We now summarize the key challenges of answering the above problem as follows:

1) **Coupled Decision Variables**. In a long-term multicast process, multicast scheduling and routing are intertwined, both exerting impact on the AoIs of destinations.
2) **Energy Constraints**. Real-world applications often impose energy constraints on the network. This introduces a trade-off between energy consumption and AoI.
3) **Hidden Graph Information**. Traditional methods are inefficient when extracting relevant graph features due to the non-Euclidean nature of graphs.
4) **NP-hardness**. Multicast routing algorithms usually fall into the realm of CO problems, which are computationally intractable for large-scale networks.

One promising solution approach to overcoming the above challenges includes Reinforcement Learning (RL) methods, which are promising for approximating the optimal solutions of NP-hard problems via learning from environments [13]. Hence, we further prompt the following question: *How should one design an RL framework to address the problem of coupled decision variables and capture the graph information of a multicast network?*

C. Solution Approach and Contributions

To answer the above questions, we summarize our solution approach and main contributions as follows:

- **Joint Multicast Scheduling and Routing Problem:** We tackle the complex task of joint multicast routing and scheduling, accounting for energy constraints and possible network dynamics. *To the best of our knowledge, this is the first work to consider the problem of minimizing AoI in joint multicast scheduling and routing.*
- **Hierarchical RL Framework.** To address challenges 1, 2 and 4, we decompose the original problem into two subtasks and introduce a hierarchical RL framework. The first subtask involves selecting destinations, while the second subtask generates multicast trees.
- **A Novel Graph Attention Network.** To overcome challenges 3 and 4, we propose a novel kind of Graph Attention Network (GAT) to extract graph information based on the attention mechanism.
- **Performance Evaluation.** We validate our approach on three datasets, including a real-world dataset called AS-733. TGMS outperforms other baselines and achieves an average AoI reduction of 62.9% while maintaining the energy consumption within the constraint.

II. System Model

A. Network Description

We consider a multicast network that operates for an infinite horizon of slotted time. See an illustrative example in Fig. 1. The network topology at time t is denoted as $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$, where $\mathcal{V}_t$ represents the set of nodes and $\mathcal{E}_t$ represents the set of undirected links. The nodes in the network can be categorized into three distinct types:

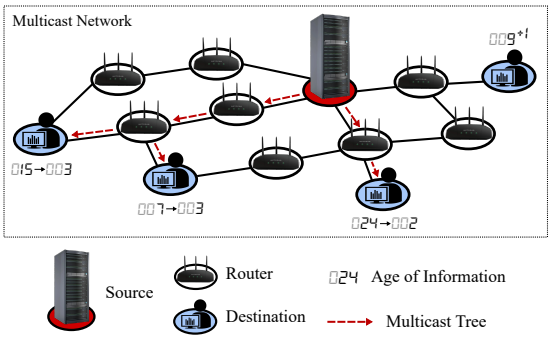

Fig. 1. An Example of a Multicast Network. The nodes are connected by links with different costs. At the beginning of each time slot, the source generates update packets, which are then forwarded to destinations by routers.

- **The Source Node.** A source node generates updates for a multicast group[1] continuously without restrictions.
- **Router Nodes.** Router nodes are responsible for forwarding update packets to destinations.
- **Destination Nodes.** Destination nodes are expected to receive update packets from the source node. The set of destinations at time t is denoted as $\mathcal{U}_t \subset \mathcal{V}_t$.

A packet can be transmitted from node i to node j if the link (i, j) exists in $\mathcal{E}_t$, which takes one time slot with an energy cost of $C_{i,j}$. The multicast process can be described as follows: at the beginning of each time slot, the source node generates multiple update packets, which are then transmitted between router nodes, eventually reaching the destination nodes. Note that packets traveling through different transmission paths may not arrive at the destination simultaneously. To analyze the AoI of destinations, we initially define the AoI of update packets. Let $\mathcal{P}_t = \{0, 1, \dots, p, \dots\}$ denotes the set of packets at time t, the AoI of packet p can be defined as:

$$\hat{A}_p(t) = t - t_p, \forall t \geq t_p, \tag{1}$$

where t_p denotes the time when packet p is generated. That is, $\hat{A}_p(t)$ grows linearly with time. Suppose that each destination can receive only one packet during a single time slot. The AoI of a destination $u \in \mathcal{U}_t$ is defined as:

$$A_u(t) = \begin{cases} \hat{A}_p(t), & \text{if } d_{u,p}(t) = 1, \\ A_u(t-1) + 1, & \text{otherwise}, \end{cases} \tag{2}$$

where $d_{u,p}(t)$ is an indicator that represents whether packet p arrives at destination u at time t. If packet p arrives at destination u at time t, $d_{u,p}(t) = 1$; otherwise, $d_{u,p}(t) = 0$. As shown in Fig. 2, if a destination u receives a packet at time t, $A_u(t)$ is updated to be the AoI of the received packet at that time. Otherwise, $A_u(t)$ grows linearly with time.

The AoIs of destinations are closely intertwined with the routing decisions. In this paper, we adopt the use of multicast

[1]This scenario can also be extended to accommodate multiple multicast groups, where each group has its source node.

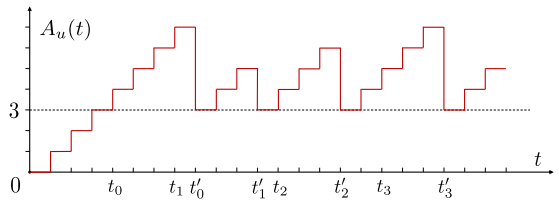

Fig. 2. An Example of the AoI. t_k denotes the time when the k-th packet is generated while t'_k denotes the time when it arrives. The AoI of destination u is updated to be the AoI of the received packet (3 in this case) at time t'_k, otherwise, it grows linearly with time.

trees to represent routing decisions as commonly done in classical multicast routing problems [9]. A tree is defined as a connected acyclic undirected graph. Accordingly, we define a multicast tree $\mathcal{T}_t$ of network $\mathcal{G}_t$ as $\mathcal{T}_t = \{\mathcal{V}_t^{\mathcal{T}}, \mathcal{E}_t^{\mathcal{T}}\}$, where $\mathcal{V}_t^{\mathcal{T}} \subseteq \mathcal{V}_t$ denotes the set of included nodes and $\mathcal{E}_t^{\mathcal{T}} \subseteq \mathcal{E}_t$ denotes the set of links. To establish the relation between the AoIs and multicast trees, we refer to the following lemma:

Lemma 1 ([14])**.** Any two vertices in a tree can be connected by a unique simple path.

Considering the multicast process described and the store-and-forward mechanism [15], we can derive that the time required for a packet to reach destination u is equal to the number of hops between the source and destination u in a multicast tree $\mathcal{T}_t$, denoted as $h_{\mathcal{T}_t}(u)$. Then Eq. (2) can be rewritten as:

$$A_u(t + h_{\mathcal{T}_t}(u)) = \begin{cases} h_{\mathcal{T}_t}(u), & \text{if } u \in \mathcal{V}_t^{\mathcal{T}}, \\ A_u(t + h_{\mathcal{T}_t}(u) - 1) + 1, & \text{otherwise.} \end{cases} \tag{3}$$

Here, $A_u(t + h_{\mathcal{T}_t}(u))$ will be updated to exactly $h_{\mathcal{T}_t}(u)$ when a packet generated at t arrives at destination u at time $t + h_{\mathcal{T}_t}(u)$, given that u is included in the multicast tree $\mathcal{T}_t$. Thus, $A_u(t)$ is influenced by two factors: (i) the frequency of which destination u is updated; (ii) the AoI of arrived packets, i.e., $h_{\mathcal{T}_t}(u)$. Both factors are determined by multicast trees. Therefore, a proper multicast tree is essential for optimizing AoIs. In addition, some scenarios bring new influence factors, which are discussed in the following sections.

B. Network Dynamics

A real network usually exhibits dynamic behavior. For instance, in wireless networks, there are topology changes from node mobility or link instability. We assume that the network topology at time t is generated by a random process, which is independent of the past. This assumption is reasonable in many scenarios (e.g., [16]). We first define a link indicator $\sigma(e, t)$ as follows:

$$\sigma(e, t) = \begin{cases} 1, & \text{if link } e \in \mathcal{E}_t, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

For a network, the distribution of links at time t is denoted as $\sigma(t) \in \Xi$, where Ξ denotes all possible network topologies. $\sigma(t)$ evolves according to a stationary ergodic process:

$$\sum_{\sigma \in \Xi} p(\sigma) = 1, p(\sigma) > 0, \forall \sigma \in \Xi. \tag{5}$$

The statistical property of the process above depends on specific network scenarios, which are often difficult to obtain. We assume that the process is unknown in priority. In the subsequent sections, we will show how our proposed approach handles such dynamic scenarios naturally by making decisions on discrete time slots in model design and implementation.

C. Energy-efficient Multicast Scheduling

In real networks, energy is a critical resource that needs to be managed efficiently. Here we propose a scheme to balance the energy consumption and AoIs, which we refer to as "multicast scheduling" in this paper. Specifically, consider a long-term energy budget denoted by W which constrains the average energy consumption. Let $C(\mathcal{T}_t)$ denote the energy consumption of a multicast tree $\mathcal{T}_t$, our goal is to find multicast trees that minimize the AoIs while satisfying the energy constraint. Formally, we formulate this problem as follows:

$$\textbf{OP:} \quad \min_{\boldsymbol{\mathcal{T}}} \quad \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}_{\boldsymbol{\mathcal{T}}}\left[\sum_{t=0}^{T}\sum_{u\in\mathcal{U}_t}\omega_u A_u(t)\right], \tag{6a}$$

$$\text{s.t.} \quad \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\boldsymbol{\mathcal{T}}}\left[\sum_{t=0}^{T} C(\mathcal{T}_t)\right] \le W, \tag{6b}$$

where $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_t | t = 0, 1, \cdots, T\}$ is a sequence of multicast trees generated over time, $\omega_u \in (0, 1)$ represents the importance of a destination $u \in \mathcal{U}_t$. In other words, if we view the above problem as a sequential decision-making problem, we aim to find a policy π to determine $\mathcal{T}_t$ as a function of the network state $\{\mathcal{T}_\tau, \mathcal{G}_\tau, A(\tau) | 0 \le \tau \le t-1\}$. Given the set of destinations $\mathcal{U}_t$, define the solution space of multicast trees including $\mathcal{U}_t$ as $\Omega(\mathcal{U}_t)$, the problem above can be rewritten as:

$$\begin{aligned} \textbf{DP:} \quad \max_{\lambda\ge 0} \quad \inf_{\boldsymbol{\mathcal{U}'},\boldsymbol{\mathcal{T}}} \Bigg(& \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\boldsymbol{\mathcal{T}}}\left[\sum_{t=0}^{T}\sum_{u\in\mathcal{U}_t}\omega_u A_u(t) + \right. \\ & \left. \lambda\Big(\sum_{t=0}^{T} C(\mathcal{T}_t) - TW\Big)\right]\Bigg), \\ \text{s.t.} \quad & \mathcal{T}_t \in \Omega(\mathcal{U}'_t), \forall t. \end{aligned} \tag{7}$$

where λ is the Lagrangian multiplier corresponding to Eq. (6b). The large and discrete solution space and the mutual influence between decision variables make the problem challenging to solve. Therefore, we reformulate the original problem into two subproblems.

III. Problem Reformulation

Our main approach is to decompose Eq. (6) into two subproblems. The first subproblem involves identifying the set of destinations that should be updated at each time slot

and is referred to as the scheduling subproblem. The second subproblem entails finding an optimal multicast tree for the selected destinations and is known as the tree-generating subproblem. Each of them can be formulated as an MDP and solved by RL methods.

A. Problem Decomposition

The two subproblems are formulated as follows:

Definition 1 (**Scheduling Subproblem**). Given a network $\mathcal{G}_t$, find a sequence $\boldsymbol{\mathcal{U}'} = \{\mathcal{U}'_t | t = 0, 1, \ldots, T\}$ of destination sets $\mathcal{U}'_t \in \mathcal{U}_t$ to:

$$\textbf{P1:} \quad \max_{\lambda \geq 0} \inf_{\boldsymbol{\mathcal{U}'}} \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} g(\lambda, \mathcal{U}'_t) \tag{8}$$

where $g(\lambda, \mathcal{U}'_t)$ is given by the optimal values of the following tree-generating subproblem.

Definition 2 (**Tree-generating Subproblem**). Given a network $\mathcal{G}_t$ and a set of destinations $\mathcal{U}'_t \in \mathcal{U}_t$, find a multicast tree $\mathcal{T}_t = \{\mathcal{V}_t^{\mathcal{T}}, \mathcal{E}_t^{\mathcal{T}}\}$ that:

$$\begin{aligned} \textbf{P2:} \quad \min_{\mathcal{T}_t} \Bigg(\lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{\mathcal{T}_t} \Bigg[\sum_{t=0}^{T} \sum_{u \in \mathcal{U}_t} \omega_u A_u(t) + \\ \lambda \Big(\sum_{t=0}^{T} C(\mathcal{T}_t) - TW \Big) \Bigg] \Bigg), \\ \text{s.t.} \quad \mathcal{T}_t \in \Omega(\mathcal{U}'_t). \end{aligned} \tag{9}$$

We can observe the equivalence between two subproblems and the original problem. In addition, the problem **P1** and **P2** can be both regarded as sequential decision-making problems, which can be formulated as MDPs. We first focus on problem **P1** and define MDP $\mathcal{M}_1 = \{\mathcal{S}_1, \mathcal{A}_1, f_1, r_1\}$ as follows:

- **States:** The state s_t is defined as:

$$s_t = \{\mathcal{G}_t, \mathbf{o}_t\}, s_t \in \mathcal{S}_1, \tag{10}$$

  where $\mathbf{o}_t = \{\mathbf{x}_t, \mathbf{e}_t\}$ denotes the features at time slot t, including node features $\mathbf{x}_t$ and link features $\mathbf{e}_t$.
- **Actions:** The action is a set of destinations, i.e.:

$$a_t = \{\mathcal{U}'_t | \mathcal{U}'_t \subseteq \mathcal{U}_t\}, a_t \in \mathcal{A}_1. \tag{11}$$

- The transition function f_1 is unknown in priority.
- **Rewards:** To assess the network's quality at time slot t, we introduce a quality function $q_1(s_t)$:

$$q_1(s_t) = -\sum_{u \in \mathcal{U}_t} \omega_u A_u(t) - \lambda(C(\mathcal{G}_t) - W), \tag{12}$$

  the reward function $r_1 : \mathcal{S}_1 \times \mathcal{A}_1 \to \mathbb{R}$ is defined as:

$$r_1(s_t, a_t) = q_1(s_t) - q_1(s_{t-1}). \tag{13}$$

Remark: By discretizing time, we can obtain the network state at each time slot and make individual decisions. Therefore, we naturally address the challenge of network dynamics.

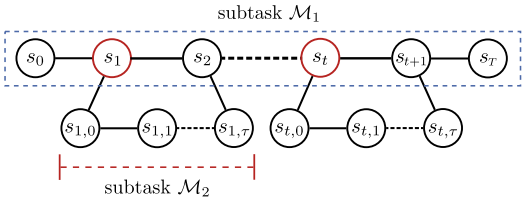

Fig. 3. Relationship between two Subtasks. $\mathcal{M}_1$ is responsible for selecting destinations, which is utilized by $\mathcal{M}_2$ to generate multicast trees.

B. Tree-generating MDP

To tackle problem **P2**, notice that a multicast tree is composed of nodes and links. To approach this, we initiate with an empty set and incrementally add nodes and links in a virtual timescale. Specifically, we introduce a virtual timescale τ and a partial solution $\mathcal{P}_\tau = \{\mathcal{V}_\tau^{\mathcal{P}}, \mathcal{E}_\tau^{\mathcal{P}}\}$ with a source node at $\tau = 0$. An MDP $\mathcal{M}_2 = \{\mathcal{S}_2, \mathcal{A}_2, f_2, r_2\}$ is defined as follows:

- **States:** The state s_τ consists of network features and a partial solution $\mathcal{P}_\tau$, defined as [2]:

$$s_\tau = \{\mathcal{G}_t, \mathbf{o}_t, \mathcal{P}_\tau\}, s_\tau \in \mathcal{S}_2. \tag{14}$$

  When the selected destinations $\mathcal{U}'_t$ are covered by $\mathcal{P}_\tau$, s_τ will be a terminal state.
- **Actions:** The action a_τ is a neighbor of $\mathcal{P}_\tau$, defined as:

$$a_\tau = v, v \in \mathcal{N}(\mathcal{P}_\tau), a_\tau \in \mathcal{A}_2, \tag{15}$$

  where $\mathcal{N}(\mathcal{P}_\tau)$ represents the neighbor nodes of $\mathcal{P}_\tau$. A link $(v^*_\tau, a_\tau), v^*_\tau \in \mathcal{P}_\tau$ will be uniquely selected as described in section IV and added to $\mathcal{P}_\tau$ along with node a_τ.
- The transition function f_2 is unknown in priority.
- **Rewards:** The quality function $q_2(s_\tau)$ can be defined as:

$$q_2(s_\tau) = \sum_{u \in \mathcal{U}'_t \cap \mathcal{V}_\tau^{\mathcal{P}}} \frac{\omega_u A_u(t)}{h_{\mathcal{P}_\tau}(u)} - \lambda(C(\mathcal{P}_\tau) - W), \tag{16}$$

  where $h_{\mathcal{P}_\tau}(u)$ is the number of hops between the source and destination u in $\mathcal{P}_\tau$. Subsequently, the reward function $r_2 : \mathcal{S}_2 \times \mathcal{A}_2 \to \mathbb{R}$ is defined as:

$$r_2(s_\tau, a_\tau) = q_2(s_\tau) - q_2(s_{\tau-1}). \tag{17}$$

Therefore, the original problem can be solved by solving $\mathcal{M}_1$ and $\mathcal{M}_2$ sequentially, which is illustrated in Fig. 3. To achieve this, we propose an algorithm called Tree Generator-based Multicast Scheduling (TGMS).

IV. Tree Generator-based Multicast Scheduling

Our proposed approach consists of a tree generator and a scheduler, both utilizing graph embedding methods and DRL techniques.

[2] s_τ is actually the abbreviation of $s_{t,\tau}$.

A. Graph Embedding Methods

Having a comprehensive understanding of the graph topology is essential to make informed decisions in network optimization problems. Graph embedding methods (e.g., GNNs) have demonstrated their efficacy in various CO problems [17]. Some studies focus on the attention mechanism in GNNs, which allows nodes to selectively aggregate information from neighbors (e.g., [18]). Based on these observations, we propose a novel GAT with guaranteed contraction mapping properties to extract graph information, which is defined as follows:

$$\phi(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{a}^{\mathrm{T}}\text{LeakyReLU}(\mathbf{W}_1(\mathbf{h}_i + \mathbf{h}_j) + \mathbf{W}_2\mathbf{e}_{i,j}), \tag{18a}$$

$$\alpha_{ij} = \frac{\exp(\phi(\mathbf{h}_i, \mathbf{h}_j))}{\sum_{k\in\mathcal{N}_i\cup\{i\}}\exp(\phi(\mathbf{h}_i, \mathbf{h}_k))}, \tag{18b}$$

$$\begin{aligned} f_{\text{GAT}}(\mathbf{h}_i, \mathbf{x}) = \frac{1}{\|\mathbf{W}_1\|}&(\alpha_{ii}(\mathbf{W}_1\mathbf{h}_i + \mathbf{W}_3\mathbf{x}_i) \\ &+ \sum_{j\in\mathcal{N}_i}\alpha_{ij}(\mathbf{W}_1\mathbf{h}_j + \mathbf{W}_3\mathbf{x}_j)), \end{aligned} \tag{18c}$$

Here, $\mathbf{h}_i \in \mathbb{R}^d$ denotes the embedding vector of node i, $\mathbf{x}_i$ denotes the node features of i, $\mathbf{e}_{i,j}$ denotes the link features of (i, j) and LeakyReLU$(\cdot)$[3] is an activation function. The new representation of node i is obtained by a weighted sum of neighbors' node embeddings using α_{ij}. Hence, we effectively disseminate information through the graph and aggregate node embeddings.

B. Model Design

Recall that the action space of $\mathcal{M}_1$ (see Eq. (11)) is excessively large[4], making it impractical for RL algorithms to efficiently explore. To tackle this challenge, we arrange the destinations $\mathcal{U}_t$ in descending order of their weighted AoIs, and then select a part of the destinations. However, the size of $\mathcal{U}_t$ varies across different networks, rendering it difficult to determine a fixed value. To overcome this issue, we devise an agent with a continuous action space. Specifically, we utilize a model to predict the mean and standard deviation of a Gaussian distribution, which are then employed to sample the fraction of destinations to be selected. The forwarding process of the scheduler can be summarized as follows:

$$\mathbf{H}_t^{(l+1)} = f_{\text{GAT}}(\{\mathbf{h}_{t,i}^{(l)}\}_{i\in\mathcal{V}_t}), \tag{19a}$$

$$\tilde{\mathbf{h}}_t = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\mathbf{h}_t^{(L)}, \tag{19b}$$

$$\mu = \mathbf{W}_1\text{LeakyReLU}(\mathbf{W}_2\tilde{\mathbf{h}}_t), \tag{19c}$$

$$\sigma = \mathbf{W}_3\text{LeakyReLU}(\mathbf{W}_4\tilde{\mathbf{h}}_t), \tag{19d}$$

$$\pi_1(a_t|s_t) = \mathcal{N}(\mu, \sigma), \tag{19e}$$

$$V_1(s_t) = \mathbf{W}_5\text{LeakyReLU}(\mathbf{W}_6\tilde{\mathbf{h}}_t). \tag{19f}$$

The initial graph embedding $\mathbf{h}_0$ is generated by combining the graph topology and features, which is updated by the GAT defined by Eq. (18a)-(18c). Following this, the outputs are passed through a pooling layer serving to aggregate global information (see Eq. (19b)). Finally, the resulting graph embeddings are fed into two distinct heads: (i) a policy head responsible for predicting the mean and standard deviation of a Gaussian distribution (see Eq. (19c)-(19e)); and (ii) a critic head tasked with predicting the expected value of the current state (see Eq. (19f)). For the tree generator, we employ similar graph embedding methods as those utilized in the scheduler (see (19a)-(19b))). The difference lies in the heads as follows:

$$\pi_2(a_\tau|s_\tau) = \log\text{softmax}(\mathbf{W}_1'\sigma(\mathbf{W}_2'\mathbf{H}_\tau^{(L)})), \tag{20a}$$

$$V_2(s_\tau) = \mathbf{W}_3' f_{\text{LeakyReLU}}(\mathbf{W}_4'\tilde{\mathbf{h}}_\tau), \tag{20b}$$

where the policy $\pi_2(a_\tau|s_\tau)$ is masked to ensure valid actions. One remaining question is that when an action a_τ is sampled from $\pi_2(a_\tau|s_\tau)$, it is possible for a_τ to have multiple links with $\mathcal{P}_\tau$. To address this, we select the link with the minimum cost from the set of candidate links, i.e.:

$$(v_\tau^*, a_\tau) = \underset{v\in\mathcal{P}_\tau,(v,a_\tau)\in\mathcal{E}_\tau^{\mathcal{P}}}{\arg\min} C_{v,a_\tau}. \tag{21}$$

This approach effectively reduces the complexity of the tree generator while constraining the cost of $\mathcal{P}_\tau$. Importantly, it does not alter the solution space of **P2**. The system architecture of our approach is illustrated in Fig. 4.

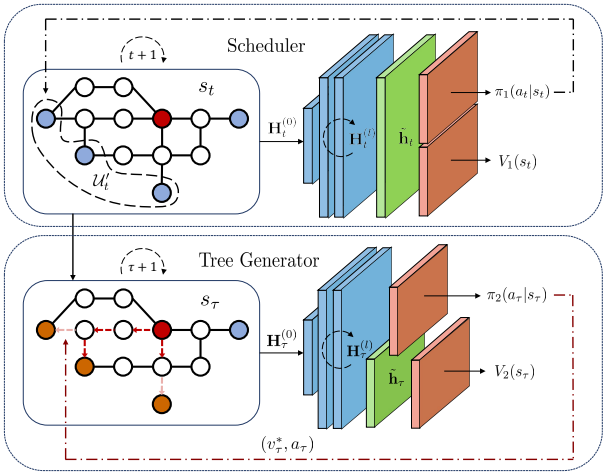

Fig. 4. The System Architecture of TGMS. A scheduler is performed to select a set of destinations, which are utilized by the tree generator to generate a multicast tree.

V. Performance Evaluation

Due to the lack of existing research on the proposed problem, we conduct extensive experiments to evaluate the performance of our approach. To validate the generalization ability of our approach, we consider three datasets of different graph topologies as follows. One of the datasets is called AS-733, which is collected from the University of Oregon Route Views Project [19]. We randomly select 240 graphs for training and 60 graphs for testing, where the testing graphs are unseen during training. Each graph will be trained or tested

[3]LeakyReLU$(x) = \max(\alpha x, x)$ where α is a hyparamater.
[4]The dimension of $|\mathcal{A}_2|$ is $2^{|\mathcal{U}_t|}$.

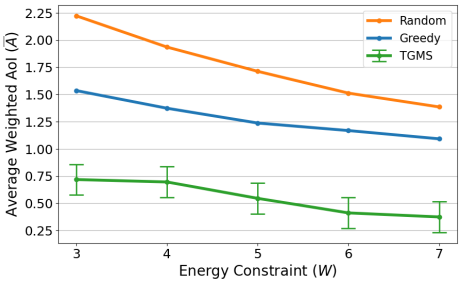

Fig. 5. Average Weighted AoI of AS-733 under $\mathbf{W}_1$.

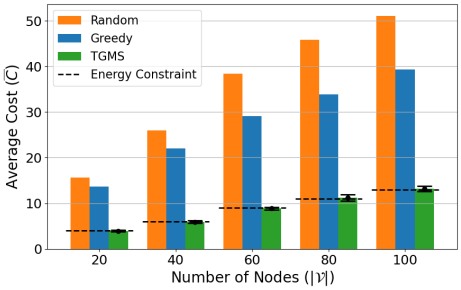

Fig. 6. Energy Consumption of AS-733 under $\mathbf{W}_2$.

for 100 time slots. We randomly select 5 seeds and record the results with the mean and standard deviation. The following algorithms are considered as baselines:

- **Random:** A fraction m of nodes are randomly selected as destinations. When generating a multicast tree, we randomly select a valid node with its minimum-cost link.
- **Greedy:** A 0.5 fraction of sorted destinations are greedily selected. When generating a multicast tree, we greedily select a valid link with minimum cost.

Consider a set of energy constraints $\mathbf{W}_1 = \{3, 4, 5, 6, 7\}$ for graph size $|\mathcal{V}| = 60$, we compare the average weighted AoI under different energy constraints. From Fig. 5, we conclude that TGMS has a superior performance compared to the baselines. When restricting energy consumption, TGMS can achieve a lower weighted AoI. Specifically, TGMS reduces the average weighted AoI by 57.1% and 68.7% compared to the greedy and random baselines, respectively.

Next, we evaluate the energy consumption under different energy constraints on different sizes of graphs. Consider another set of energy constraints $\mathbf{W}_2 = \{4, 6, 9, 11, 13\}$ for graph sizes $|\mathcal{V}| = \{20, 40, 60, 80, 100\}$, respectively. We maintain a similar weighted AoI for all algorithms and compare the energy consumption. The results are shown in Fig. 6. We observe that TGMS achieves a lower energy consumption compared to the baselines. Specifically, TGMS reduces the energy consumption by 75.7% and 69.3% compared to the random and greedy baselines, respectively. *The additional experiments can be found in the appendix of [20].*

VI. Conclusion

In this paper, we have proposed a novel hierarchical RL architecture including a GAT-based graph embedding method. It concludes with two agents: (i) a scheduler that selects a set of destinations while meeting an average power constraint, and (ii) a tree generator for generating multicast trees. We compare with several baselines and confirm that TGMS outperforms them in terms of AoI reduction and energy efficiency.

References

[1] X. Jiang, F. R. Yu, T. Song, and V. C. Leung, "A survey on multi-access edge computing applied to video streaming: Some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 871–903, 2021.

[2] L. Yin, J. Gui, Z. Zeng *et al.*, "Improving energy efficiency of multimedia content dissemination by adaptive clustering and d2d multicast," *Mobile Information Systems*, vol. 2019, 2019.

[3] B. Quinn and K. Almeroth, "Ip multicast applications: Challenges and solutions," Tech. Rep., 2001.

[4] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2731–2735.

[5] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[6] J. Li, Y. Zhou, and H. Chen, "Age of information for multicast transmission with fixed and random deadlines in iot systems," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8178–8191, 2020.

[7] S. F. Lindström, M. Wetterberg, and N. Carlsson, "Cloud gaming: A qoe study of fast-paced single-player and multiplayer gaming," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2020, pp. 34–45.

[8] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "Qoe management of multimedia streaming services in future networks: a tutorial and survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 526–565, 2019.

[9] C. A. Oliveira and P. M. Pardalos, "A survey of combinatorial optimization problems in multicast routing," *Computers & Operations Research*, vol. 32, no. 8, pp. 1953–1981, 2005.

[10] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[11] M. Xie, J. Gong, X. Jia, and X. Ma, "Age and energy tradeoff for multicast networks with short packet transmissions," *IEEE Transactions on Communications*, vol. 69, no. 9, pp. 6106–6119, 2021.

[12] I. Ljubić, "Solving steiner trees: Recent advances, challenges, and perspectives," *Networks*, vol. 77, no. 2, pp. 177–204, 2021.

[13] M. Kim, J. Park *et al.*, "Learning collaborative policies to solve np-hard routing problems," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 418–10 430, 2021.

[14] West and D. Brent, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[15] X. Lin and L. M. Ni, "Multicast communication in multicomputer networks," *IEEE transactions on Parallel and Distributed Systems*, vol. 4, no. 10, pp. 1105–1117, 1993.

[16] A. Sinha, L. Tassiulas, and E. Modiano, "Throughput-optimal broadcast in wireless networks with dynamic topology," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2016, pp. 21–30.

[17] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[18] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[19] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 177–187.

[20] Y. Zhang, G. Liao, S. Cao, N. Yang, and M. Zhang, "Age-minimal multicast by graph attention reinforcement learning," https://arxiv.org/abs/2404.18084, 2024.

[21] S. Mukherjee, F. Bronzino, S. Srinivasan, J. Chen, and D. Raychaudhuri, "Achieving scalable push multicast services using global name resolution," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[22] Q. Yu, H. Wan, X. Zhao, Y. Gao, and M. Gu, "Online scheduling for dynamic vm migration in multicast time-sensitive networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3778–3788, 2019.

APPENDIX

A. Experiment Setting

Due to the lack of existing datasets for our problem, we built an environment for training and testing, where three datasets of different graph topologies are considered as follows:

- **ER-Graphs:** The Erdos-Renyi (ER) random graphs are a type of graphs where each pair of nodes is connected with a fixed probability p. We set the link connection probability $p = 6/|\mathcal{V}|$ (e.g., in [21]).
- **BA-Graphs:** In Barabasi-Albert (BA) graphs, each newly introduced node connects to m pre-existing nodes. This connection process is governed by a probability that is directly proportional to the number of links the existing nodes already possess. We set $m = 2$ in our experiment (e.g., in [22]).
- **AS-733:** The Autonomous Systems (AS)-733 dataset is a real-world dataset collected from the University of Oregon Route Views Project [19]. It contains 733 abstracted graphs of Autonomous Systems.

For each graph, we randomly select a source node and a fraction of destinations $\delta_d = 0.3$. Subsequently, We create some initial node features and link features on the above network topologies. We randomly assign a weight $\omega_u \in (0, 1)$ and an initial AoI $A_u(0) \in [1, 5]$ for every destination $u \in \mathcal{U}_0$. The sum of weights $\sum_u \omega_u$ is ensured to be 1. The energy cost of each link is randomly assigned in the range $(0, 1)$.

The unit of training interval is the time slot. All modules are trained from scratch for 100 time slots for each graph from the training dataset. The model is tested for 100 time slots per graph of the testing dataset. We use PyGraph 2.4.0 to implement TGMS, which is trained on NVIDIA GeForce RTX 3090 and tested on AMD EPYC 7763 CPU @1.50GHz with 64 cores under Ubuntu 20.04.6 LTS.

B. Additional Experiments

We conduct additional experiments on various datasets over different graph sizes. Specifically, we consider graphs with sizes $|\mathcal{V}| = \{20, 40, 60, 80, 100\}$. The results are shown in Table I. We find that TGMS can achieve a lower average weighted AoI compared with other baselines, while the energy consumption is also lower.

C. Insights of Reward Functions

Here we provide some insights into the reward functions we adopted in our formulated MDPs. We start by calculating the cumulative reward of $\mathcal{M}_1$:

$$\begin{aligned} R_1 &= \sum_{t=0}^{T} \gamma^{t-1} r_1(s_t, a_t) \\ &= r_1(s_0, a_0) + \sum_{t=1}^{T} \gamma^t (q_1(s_t) - q_1(s_{t-1})) \\ &\overset{\gamma=1}{=\!=} r_1(s_0, a_0) + q_1(s_T) - q_1(s_0) \\ &= -\sum_{u \in \mathcal{U}_t} \omega_u A_u(T) - \lambda(C(\mathcal{T}_T) - W), \end{aligned} \tag{22}$$

$\lvert\mathcal{V}\rvert$ \ DS		ER	BA	AS
20	$\overline{A}$	0.495	0.900	0.409
	$\overline{C}$	2.315	3.761	2.278
	W	2	4	2
40	$\overline{A}$	0.288	0.306	0.378
	$\overline{C}$	6.807	8.368	9.218
	W	6	9	9
60	$\overline{A}$	0.249	0.215	0.257
	$\overline{C}$	8.084	15.434	18.595
	W	8	15	18
80	$\overline{A}$	0.171	0.140	0.573
	$\overline{C}$	11.821	22.337	17.789
	W	12	23	18
100	$\overline{A}$	0.188	0.099	0.158
	$\overline{C}$	18.034	22.797	22.622
	W	18	23	20

TABLE I
TGMS ALGORITHM

where we set $q_1(s_0) = r_1(s_0, a_0)$ and assume the discount factor $\gamma = 1$. Comparing the RHS of Eq. (22) with problem **DP**, we observe that maximizing R_1 is similar to solving problem **DP**. The difference is that problem **DP** aims to minimize a long-term objective, which is naturally decomposed as the reward function r_1.

To understand the reward function of $\mathcal{M}_2$, we first analyze how much a multicast tree $\mathcal{T}_t$ can reduce the AoI of a destination. Denote $\Delta A_u(t, t + h_{\mathcal{T}_t}(u))$ as the AoI reduction of destination u during time $[t, t + h_{\mathcal{T}_t}(u)]$, we have:

$$\begin{aligned}
&\Delta A_u(t, t + h_{\mathcal{T}_t}(u)) \\
&= A_u^+(t, t + h_{\mathcal{T}_t}(u)) - A_u^-(t, t + h_{\mathcal{T}_t}(u)) \\
&= \sum_{k=0}^{h_{\mathcal{T}_t}(u)} (A_u(t) + k) - \left(\sum_{k=0}^{h_{\mathcal{T}_t}(u)-1} (A_u(t) + k) + h_{\mathcal{T}_t}(u) \right) \\
&= A_u(t),
\end{aligned} \tag{23}$$

where $A_u^+(t, t+h_{\mathcal{T}_t}(u))$ and $A_u^-(t, t+h_{\mathcal{T}_t}(u))$ denotes the AoI of destination u during time $[t, t + h_{\mathcal{T}_t}(u)]$ before and after the multicast tree $\mathcal{T}_t$ is generated, respectively. That means whatever the multicast tree is, the AoI of a destination will be reduced by $A_u(t)$ after $h_{\mathcal{T}_t}(u)$ time slots. Therefore, the mean AoI reduction of a destination u during time $[t, t + h_{\mathcal{T}_t}(u)]$ is:

$$\frac{1}{h_{\mathcal{T}_t}(u)} \Delta A_u(t, t + h_{\mathcal{T}_t}(u)) = \frac{A_u(t)}{h_{\mathcal{T}_t}(u)} \tag{24}$$

Then, we calculate the cumulative reward of $\mathcal{M}_2$ as:

$$\begin{aligned}
R_2 &= \sum_{t=0}^{T} \gamma^{t-1} r_2(s_t, a_t) \\
&= r_2(s_0, a_0) + \sum_{t=1}^{T} \gamma^t (q_2(s_t) - q_2(s_{t-1})) \\
&\overset{\gamma=1}{=\!=} r_2(s_0, a_0) + q_2(s_T) - q_2(s_0) \\
&= \sum_{u \in \mathcal{U}_t' \cap \mathcal{V}_\tau^{\mathcal{P}}} \frac{\omega_u A_u(t)}{h_{\mathcal{P}_\tau}(u)} - \lambda(C(\mathcal{P}_\tau) - W).
\end{aligned} \tag{25}$$

Comparing Eq. (24) and the RHS of Eq. (25), we claim that maximizing R_2 is similar to maximizing the average weighted AoI reduction of destinations $\mathcal{U}_t' \cap \mathcal{V}_\tau^{\mathcal{P}}$ during time $[t, t+h_{\mathcal{P}_\tau}(u)]$. Note that solely maximizing R_2 may not equal solving problem **DP**. However, by combining two MDPs, we can obtain a solution that is close to the optimal solution of problem **DP**.

D. Contraction Mapping of GAT

For the proposed GAT, we have the following theorem:

Theorem 1. For any undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, given a mapping f_{GAT} defined in Eq. (18a)-(18c), if the attention coefficients α_{ij} are symmetric, i.e., $\alpha_{ij} = \alpha_{ji}$. Then $f_{\text{GAT}}(\cdot, \mathbf{x})$ is a contraction mapping for any initial node embeddings, i.e.:

$$d(f_{\text{GAT}}(\mathbf{H}, \mathbf{x}), f_{\text{GAT}}(\mathbf{H}', \mathbf{x})) \leq d(\mathbf{H}, \mathbf{H}'), \tag{26}$$

where $\mathbf{H} = \{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{|\mathcal{V}|}\}$ is the matrix of node embeddings, d is a distance metric with respect to $\mathbf{H}$, defined as $d(\mathbf{H}, \mathbf{H}') = \| \sum_{i \in \mathcal{V}} (\mathbf{h}_i - \mathbf{h}_i') \|$.

Proof. Let node embeddings $\mathbf{H} = \{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{|\mathcal{V}|}\}$ and $\mathbf{H}' = \{\mathbf{h}_0', \mathbf{h}_1', \ldots, \mathbf{h}_{|\mathcal{V}|}'\}$. From Eq. (**??**), we have:

$$\begin{aligned}
&d(f_{\text{GAT}}(\mathbf{H}, \mathbf{x}), f_{\text{GAT}}(\mathbf{H}', \mathbf{x})) \\
&= \| \sum_{i \in \mathcal{V}} (f_{\text{GAT}}(\mathbf{h}_i, \mathbf{x}) - f_{\text{GAT}}(\mathbf{h}_i', \mathbf{x})) \| \\
&= \| \sum_{i \in \mathcal{V}} \frac{1}{\|\mathbf{W}_1\|} (\alpha_{ii} (\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_3 \mathbf{x}_i) \\
&\quad + \sum_{j \in \mathcal{N}_i} \alpha_{ij} (\mathbf{W}_1 \mathbf{h}_j + \mathbf{W}_3 \mathbf{x}_j) - \alpha_{ii} (\mathbf{W}_1 \mathbf{h}_i' + \mathbf{W}_3 \mathbf{x}_i) \\
&\quad - \sum_{k \in \mathcal{N}_i} \alpha_{ik} (\mathbf{W}_1 \mathbf{h}_k' + \mathbf{W}_3 \mathbf{x}_k)) \| \\
&= \frac{1}{\|\mathbf{W}_1\|} \| \sum_{i \in \mathcal{V}} (\alpha_{ii} \mathbf{W}_1 (\mathbf{h}_i - \mathbf{h}_i') + \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_1 \mathbf{h}_j \\
&\qquad - \sum_{k \in \mathcal{N}_i} \alpha_{ik} \mathbf{W}_1 \mathbf{h}_k' \| \\
&= \frac{1}{\|\mathbf{W}_1\|} \| \mathbf{W}_1 \sum_{i \in \mathcal{V}} (\alpha_{ii} (\mathbf{h}_i - \mathbf{h}_i') + \sum_{k \in \mathcal{D}_i} \alpha_{ki} (\mathbf{h}_i - \mathbf{h}_i')) \| \\
&\overset{\alpha_{ij} = \alpha_{ji}}{=\!=\!=} \frac{1}{\|\mathbf{W}_1\|} \| \mathbf{W}_1 \sum_{i \in \mathcal{V}} (\alpha_{ii} + \sum_{k \in \mathcal{D}_i} \alpha_{ik}) (\mathbf{h}_i - \mathbf{h}_i') \| \\
&= \frac{1}{\|\mathbf{W}_1\|} \| \mathbf{W}_1 \sum_{i \in \mathcal{V}} (\alpha_{ii} + \sum_{j \in \mathcal{N}_i} \alpha_{ij}) (\mathbf{h}_i - \mathbf{h}_i') \| \\
&\leq \| \sum_{i \in \mathcal{V}} (\mathbf{h}_i - \mathbf{h}_i') \| = d(\mathbf{H}, \mathbf{H}').
\end{aligned} \tag{27}$$

Therefore, the mapping $f_{\text{GAT}}(\cdot, \mathbf{x})$ is a contraction mapping. □