Source-Coded Online Algorithm for Multicast Subgraph Construction

Tomás Lestayo Martínez*, Manuel Fernández Veiga† *(Senior Member, IEEE)*

***Abstract*—Multicast remains a fundamental mechanism for scalable content distribution, yet existing approaches face critical limitations. Traditional multicast trees suffer from path redundancy and inefficient utilization of network resources, while network coding, although capacity-achieving, incurs significant computational overhead and deployment challenges. In this paper, we introduce a source-coded multicast framework that exploits maximum-flow decomposition to construct multiple disjoint or partially overlapping paths from the source to all receivers. Our scheme incorporates a novel path redirection mechanism: when multiple overlaps occur between receiver flows, downstream paths are realigned at the first intersection, ensuring loop-free delivery while maximizing overall throughput. We develop algorithms for path construction, overlap detection, and iterative refinement of multicast subgraphs, and analyze their computational complexity. Through extensive evaluation on synthetic and real network topologies, we demonstrate that the proposed method consistently approaches the throughput of network coding with substantially lower encoding and decoding complexity, while significantly outperforming multicast tree constructions in terms of fairness, robustness to link failures, and delivery efficiency. These results position source-coded multicast as a practical and scalable solution for next-generation networks requiring high-throughput and adaptive group communication.**

***Index Terms*—Muticast, Source-Coded, Network Coding, Content Distribution Network, Random graphs.**

I. Introduction

MULTICAST communication has long been recognized as a cornerstone of scalable content delivery in modern communication networks. Applications such as IPTV, real-time video streaming, distributed computing, and large-scale data dissemination rely on the ability to efficiently deliver identical content from a single source to multiple receivers. Traditional multicast mechanisms typically construct rooted trees that span the source, ensuring minimal duplication of traffic across the network. Although effective in simple scenarios, these tree-based approaches are inherently limited by their rigidity: they often fail to fully exploit path diversity, are highly sensitive to link bottlenecks, and provide limited resilience to failures [1]–[3].

To overcome these limitations, a natural formulation is to compute multiple edge-disjoint or partially overlapping paths between the source and each receiver, thereby exploiting network diversity. This reduces to the maximum flow problem between a single source and a single sink. The classical framework is the Ford–Fulkerson method [4], later refined by Dinic's algorithm [5] and the Edmonds–Karp algorithm [6], which provide polynomial-time solutions. In this work, we adopt Edmonds–Karp as our baseline: its use of breadth-first searches to identify augmenting paths aligns well with our online design, and it performs efficiently in the unit-capacity networks considered here.

Network coding has emerged as a theoretically optimal solution, achieving the multicast capacity by allowing intermediate nodes to forward linear combinations of data symbols in transit toward the next hop [7]–[9]. However, despite its elegance, the practical deployment of network coding has been hampered by challenges such as computational overhead, synchronization requirements, and integration with forwarding infrastructures [10], [11]. These limitations motivate the search for alternatives that retain part of the performance gains of coding while preserving the simplicity of path-based forwarding.

In this work, we propose an *online source-coded multicast framework* that incrementally builds receiver flows through color-constrained breadth-first searches. Instead of recomputing maximum flows globally with Edmonds–Karp for each new receiver, our algorithm processes receivers sequentially, enforcing *color constraints* that control how paths may overlap. Each color represents a flow substream assigned to a receiver path; a receiver never reuses the same color across its paths, and any overlap with existing flows is limited to a single color. When a newly added path intersects multiple existing flows, it is redirected at the first overlap point, ensuring loop-free construction and feasibility. Crucially, these constraints align with the conditions identified in the works [12]–[14], which show that such overlap patterns remain decodable under source-only coding, without requiring in-network operations. This design eliminates global recomputation, adapts naturally to dynamic receiver arrivals, and scales efficiently with network size.

For comparison purposes, we also construct network coding–optimal multicast subgraphs by applying Edmonds–Karp independently for each receiver and merging the resulting flows. This provides a benchmark that isolates the structural advantage of coding from the computational overhead of symbol mixing, and allows us to contextualize the performance of our online framework relative to both classical trees and coding-enabled multicast.

The main contributions of this paper are summarized as follows:

1) We introduce an online source-coded multicast framework that incrementally constructs edge-disjoint or color-constrained overlapping paths from the source to receivers.
2) We formalize a color-constrained breadth-first search that enforces overlap consistency, ensuring feasibility and compatibility with source coding at the sender.

3) We design algorithms for online path integration and redirection at intersections, enabling multicast subgraphs to be built without global recomputation.
4) We establish a coding-optimal benchmark by running Edmonds–Karp per receiver and merging flows, providing a natural baseline to assess coding gains.
5) We evaluate the framework on synthetic and real network topologies, showing that it achieves near-optimal throughput compared to network coding with substantially lower complexity, and that it outperforms multicast trees in terms of fairness, resilience, and link utilization.

Extensive evaluations confirm that the proposed online framework offers a scalable and practical solution for content distribution. Our results highlight the potential of combining source-based flow decomposition with lightweight online integration rules, yielding a favorable trade-off between efficiency, robustness, and deployability.

The remainder of this paper is organized as follows. Section II reviews related work on multicast tree construction, network coding, and source-coded multicast. Section III introduces the system model and problem formulation. Section IV details the proposed online framework and algorithm. Section V analyzes complexity and performance properties. Section VIII presents evaluation results and benchmarks. Finally, Section IX concludes the paper.

A. Summary of notations

For clarity, we summarize the notation used throughout this paper. $G = (V, E)$ denotes a directed network graph with vertex set V and edge set E. The capacity of edge $(u, v) \in E$ is symbolized by $c(u, v)$. The letter s will generally be used to refer to the source node, and $\mathcal{R} = \{r_1, \ldots, r_m\}$ for a set of receivers. The maximum flow from s to receiver r_i is written as $f(s, r_i)$. $\mathcal{P}_i$ is the set of paths carrying flow to receiver r_i, and $\mathcal{G}_{\text{multicast}}$ is the multicast subgraph formed by the union of $\mathcal{P}_i$.

II. Related Work

The problem of efficiently delivering identical content from one source to multiple receivers has been extensively studied in the networking literature. Early solutions focused on multicast tree construction, with approaches such as shortest-path trees (SPT) and minimum Steiner trees [1], [15]–[17]. These algorithms aim to minimize the path length or the overall use of the link while maintaining a tree structure rooted at the source. Despite their simplicity and low computational cost, tree-based multicast methods suffer from inherent limitations: they rely on a single spanning structure, cannot exploit alternative disjoint paths, and exhibit poor adaptability in the presence of link failures or dynamic network conditions. Extensions such as shared trees and dynamic Steiner approximations [18] attempted to address these issues, but the fundamental constraints of tree rigidity remained.

The natural formulation of the multicast problem is closely tied to the *maximum flow problem*, originally introduced by Ford and Fulkerson [4]. This framework established the augmenting path method for determining flow capacity between a source and a sink. Subsequent refinements led to polynomial-time algorithms, notably Dinic's blocking flow approach [5] and the Edmonds–Karp algorithm, which guarantees $O(|V||E|^2)$ complexity by relying on breadth-first searches to identify augmenting paths [6]. These algorithms form the theoretical foundation for constructing flow-based multicast subgraphs, and they continue to serve as baselines for both coding and non-coding multicast strategies.

In contrast, the introduction of network coding [8], [9], [19] revolutionized the multicast problem by demonstrating that coding operations at intermediate nodes can achieve the maximum information flow simultaneously to all receivers. Network coding eliminates the bottlenecks imposed by tree-based structures and ensures optimal utilization of network capacity. Subsequent work explored practical coding strategies, including random linear coding [10], distributed implementations [11], and robust formulations under link failures [20]. Nevertheless, despite their theoretical advantages, network coding schemes face persistent deployment challenges: (i) coding/decoding overhead at intermediate and end nodes, (ii) synchronization and coordination complexity across flows, and (iii) incompatibility with existing forwarding hardware in traditional IP networks. As a result, network coding has remained primarily within theoretical and experimental domains, with limited large-scale adoption in operational networks.

To bridge the gap between tree-based multicast and fully coded approaches, several hybrid strategies have been proposed. For example, diversity-aware multicast frameworks [2] leverage multiple disjoint or partially disjoint trees to improve resilience and throughput. Overlay-based multicast systems [3], [21] exploit application-level relays to bypass limitations of native IP multicast, while multipath forwarding techniques [22] balance traffic across parallel paths to increase robustness. These methods improve performance compared to single-tree multicast, but they do not guarantee the optimal throughput achievable by network coding and often incur additional coordination complexity.

More recently, the concept of *source-coded multicast* has been introduced as a viable alternative that retains the efficiency of flow maximization while avoiding intermediate-node coding. The early work in [12] already explored forward error correction (FEC) for reliable multicast. Building on this foundation, subsequent contributions [13], [14] demonstrated how source-side processing can enable efficient dissemination by leveraging maximum-flow decomposition without relying on traditional tree construction. Their approach highlights the potential of source-only coding mechanisms to simplify deployment and improve scalability. Our work builds upon this line of research, extending it by integrating a systematic overlap management mechanism that enforces consistent convergence of receiver paths while maximizing utilization of available network capacity.

This paper further extends the source-coded paradigm by introducing *overlap-aware online path management*: when multiple overlap events occur, a redirection policy at the first intersection enforces loop-free behavior and efficient capacity utilization—addressing gaps left by prior approaches.

III. System Model

We model the communication network as a directed multigraph $G = (V, E)$, where V denotes the set of nodes and E the set of directed edges. Each directed edge $(u, v) \in E$ represents a unit-capacity transmission resource,

$$c(u, v) = 1,$$

corresponding to one unit of information per time slot. Multiple parallel edges may exist between the same ordered pair of nodes to model higher-capacity or redundant connections. The network is assumed to be lossless and error-free at the packet level; reliability and scheduling aspects are orthogonal to the throughput analysis considered here.

A single source node $s \in V$ aims to deliver identical content to a set of receivers $\mathcal{R} = \{r_1, \ldots, r_m\} \subset V$. The objective is to maximize the rate at which all receivers can simultaneously receive the content under unit-capacity constraints. For each receiver r_i, the maximum achievable flow $f(s, r_i)$ equals the maximum number of edge-disjoint directed paths connecting s and r_i. The overall multicast rate is therefore limited by the receiver with the smallest individual max-flow:

$$K = \min_i f(s, r_i), \tag{1}$$

which we refer to as the *multicast group max-flow*.

To compute the path sets $\mathcal{P}_i$ corresponding to each r_i, we rely on classical maximum-flow algorithms. The Ford–Fulkerson method [4] defines the general framework of augmenting paths, while Dinic's blocking-flow algorithm [5] and Edmonds–Karp's refinement [6] provide polynomial-time guarantees. Edmonds–Karp is particularly convenient for the unit-capacity, directed multigraphs considered here: it identifies augmenting paths using *breadth-first search (BFS)* on the residual graph, ensuring that the length of augmenting paths is non-decreasing across iterations and yielding total complexity $O(|V||E|^2)$.

The residual graph $G_f = (V, E_f)$ is initialized as $E_f \leftarrow E$, so that each directed edge instance acts as an independent unit-capacity resource. When an augmenting path uses a directed edge $e = (u, v)$, that arc is saturated and removed from E_f by setting its residual capacity to zero. The opposite direction (v, u), if present as a distinct edge, remains available unless it is also saturated. Bidirectionality is thus modeled explicitly as two separate arcs, not as a single undirected link.

The Edmonds–Karp procedure for a given receiver r_i is summarized in Algorithm 1. The algorithm repeatedly performs BFS on the residual graph to find shortest augmenting paths from s to r_i, backtracks them upon discovery, updates the residual capacities, and stores each resulting path in $\mathcal{P}_i$. Because all edges have unit capacity, every augmentation increases the flow by one unit, and all paths in $\mathcal{P}_i$ are mutually edge-disjoint.

Figure 1 illustrates a single BFS run on an example topology with intermediate nodes U, V, and W. Panel (a) shows the layered expansion of the search from s; panel (b) records the predecessor map $\pi(\cdot)$ for each discovered node; and panel (c) reconstructs the shortest augmenting path by backtracking predecessors, resulting in $s \to U \to W \to r_i$. Each connected node pair is represented by two directed arcs, one per direction; the augmenting path uses only one orientation, leaving the reverse arc available.

Algorithm 1 Edmonds–Karp for (s, r_i)

Require: Graph $G = (V, E)$; source s; receiver r_i
Ensure: Path set $\mathcal{P}_i$ of edge-disjoint $s \to r_i$ paths

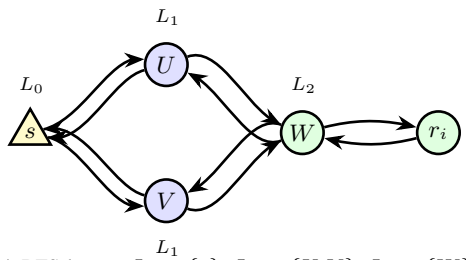
$\mathcal{P}_i \leftarrow \varnothing$; $E_f \leftarrow E$ ▷ Initialize residual graph
while true **do**
  $Q \leftarrow [\, s \,]$ ▷ BFS queue
  $\pi(v) \leftarrow \varnothing,\ \pi_{\text{edge}}(v) \leftarrow \varnothing,\ \forall v \in V$
  while $Q \neq [\,]$ **do**
    $u \leftarrow \text{head}(Q)$; $Q \leftarrow \text{tail}(Q)$
    for each edge $e = (u, v) \in E_f$ with $\pi(v) = \varnothing$ **do**
      $\pi(v) \leftarrow u$; $\pi_{\text{edge}}(v) \leftarrow e$; $Q \leftarrow Q \,\|\, [\, v \,]$
      if $v = r_i$ **then** ▷ Receiver reached
        $p \leftarrow [\,]$; $x \leftarrow r_i$ ▷ Backtrack and update E_f
        while $x \neq s$ **do**
          $e_x \leftarrow \pi_{\text{edge}}(x)$; $p \leftarrow [\, e_x \,] \,\|\, p$
          $c_f(e_x) \leftarrow 0$ ▷ Saturate edge
          $x \leftarrow \pi(x)$
        end while
        $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{p\}$
        continue ▷ Restart outer loop
      end if
    end for
  end while
  break ▷ No augmenting path found
end while
return $\mathcal{P}_i$

Each BFS identifies one additional edge-disjoint path and updates the residual graph accordingly, as shown in Fig. 2. Because edges have unit capacity, every augmentation increases the total flow by one, and the set $\mathcal{P}_i$ provides a complete decomposition of the maximum s–r_i flow into unit-capacity paths.

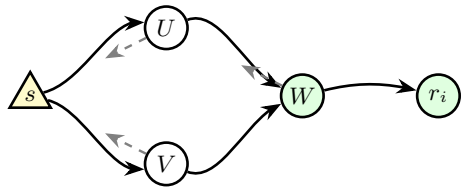
While this formulation suffices for independent receiver sessions, it does not directly address the interactions between multiple receivers when their path sets $\{\mathcal{P}_i\}$ are combined into a single multicast subgraph. Uncontrolled overlaps across receivers can compromise both capacity efficiency and the feasibility of source-only coding. To overcome these limitations, the next section extends Edmonds–Karp into an *online color-constrained BFS*, which incorporates overlap restrictions directly into path discovery. This online variant ensures that each new receiver flow is integrated consistently with previously established paths—preserving disjointness whenever possible and enforcing controlled redirection at overlaps.

IV. Proposed Framework

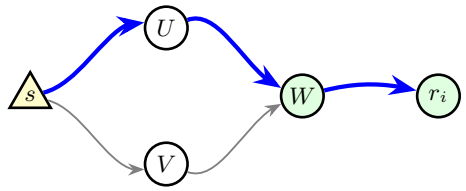
In this Section, we introduce the conceptual foundations of our online multicast framework. The goal is to exploit network path diversity while ensuring that overlaps between receiver flows remain consistent with source-only coding. Unlike classical multicast trees, which rigidly constrain all receivers to a single spanning structure, our design integrates multiple edge-disjoint or partially overlapping paths in a way that is both throughput-efficient and coding-compatible. The framework is articulated around three pillars: (i) the motivation and design principles that motivate overlap control, (ii) the notion of colors and invariants that formalize feasible overlaps, and (iii) the source-coding perspective that ensures end-to-end decodability.



a) BFS layers: $L_0 = \{s\}$, $L_1 = \{U, V\}$, $L_2 = \{W\}$



b) Predecessors π: $\pi(U) = s$, $\pi(V) = s$, $\pi(W) = U$



c) Reconstructed shortest augmenting path $s \to U \to W \to r_i$

Fig. 1: Illustrative Edmonds–Karp BFS: (a) layered expansion from s; (b) predecessor map; (c) reconstruction of the shortest $s \to r_i$ augmenting path.

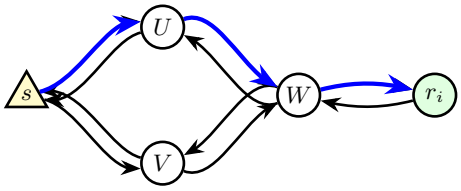
A. Motivation and Design Principles

The motivation for explicitly controlling path overlap arises from prior work on source-coded multicast [13], [14], where it was shown that only certain overlap structures between receivers are compatible with efficient source-only coding. In particular, when flows to different receivers overlap across multiple distinct paths, the independence of the delivered substreams may be compromised, preventing receivers from decoding the full content. On the other hand, if the overlap of a new receiver flow is restricted to a single preexisting path, or if it is redirected to consistently align with that path at the first intersection, the linear independence of the substreams is preserved while still compacting the multicast subgraph.

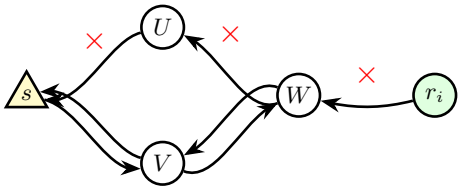
This principle guides the design of our framework:

- *Incrementality*: receivers are integrated one by one, without global recomputation.
- *Overlap control*: each new path is admitted only if it remains disjoint, or if it overlaps consistently with a single existing path.
- *Feasibility by construction*: redirection at the first overlap ensures that infeasible multi-overlaps are automatically avoided.

Together, these design principles guarantee that the resulting multicast subgraph is both loop-free and compatible with source-only coding.



a) Initial residual graph and shortest $s \to r_i$ path (blue)



b) Residual after augmentation: used forward arcs removed

Fig. 2: Residual graph before (a) and after (b) one augmentation. The arcs used by the shortest $s \to r_i$ path are removed, while unused and reverse arcs remain available for subsequent searches.

B. Colors, Zones, and Invariants

We model the multicast subgraph as $\mathcal{G}_{\text{multicast}} = (V, E_M)$, incrementally constructed as receivers are integrated. Each directed edge $e \in E_M$ is labeled with a color from a global palette $\mathcal{C}$. A color identifies a *zone*, i.e., a connected subset of edges that originated from one receiver's path and that may later be extended by subsequent receivers aligning with it.

When a new path is discovered: (i) If it is disjoint from E_M, it is assigned a fresh color (new zone); (ii) If it overlaps with exactly one existing zone, it inherits that zone's color; (iii) If it encounters multiple overlaps, it is automatically redirected at the first intersection to continue within the corresponding zone, preserving feasibility.

The algorithm preserves the following invariants:

1) **Unit-capacity feasibility**: each directed edge is saturated after being used by one path, guaranteeing that residual capacity constraints are respected.
2) **Single-path overlap consistency**: every new path may align with at most one preexisting color; multi-overlaps are prevented by redirection.
3) **Per-receiver exclusivity**: a receiver r_i assigns a distinct color to each of its own paths, ensuring independence of its incoming substreams.

C. Source-Coding Perspective

From a coding perspective, all operations remain confined to the source s. The source performs flow-splitting into independent substreams, each mapped to a unit-capacity path discovered under the color-constrained rules. Intermediate nodes simply forward packets, without any need for recoding. At the destination, each receiver r_i reconstructs the original content by combining the independent substreams delivered along its disjoint or zone-aligned paths.

This perspective highlights the advantage of the proposed framework: it achieves much of the throughput of network

coding, but without requiring complex in-network operations. Instead, the enforcement of overlap consistency through colors guarantees that the source-only coding strategy remains feasible.

In the next Section, we translate these principles into a concrete *online algorithm*. We formalize the color-constrained breadth-first search, describe the path integration process, and illustrate its operation through examples. This provides the step-by-step procedure by which the proposed framework is realized in practice.

V. Online Algorithm

Building on the system model defined in Section III and the design principles outlined in Section IV, we now present the online multicast construction algorithm in detail. Unlike classical offline approaches, which first compute full maximum-flow decompositions for each receiver and then reconcile overlaps, our method integrates overlap control directly into the path discovery stage. This design allows the multicast subgraph to be constructed incrementally as receivers arrive, without global recomputation, while guaranteeing both unit-capacity feasibility and consistency of overlaps. The algorithm relies on a *color-constrained breadth-first search (BFS)* that enforces zone alignment rules during exploration, ensuring that every augmenting path is either disjoint, aligned to a single existing zone, or redirected at its first intersection. We first describe the constrained BFS mechanism, then the integration routine for sequential receivers, and finally provide a detailed pseudocode implementation.

A. Color-Constrained BFS

For each receiver r_i, a constrained BFS is executed on the residual graph G_f. The key difference from standard Edmonds–Karp lies in how path expansions are filtered according to the active color state of the partial path. Specifically:

- If the current partial path has not touched any colored edge, it may traverse any uncolored edge freely. If it enters a colored zone, that color becomes the active color for the remainder of the path.
- Once a color c is active, only edges that are either uncolored or already marked with c may be traversed. Expansions that would mix multiple colors are forbidden.

This rule has three important consequences:

1) Paths that are entirely disjoint from E_M are discovered unchanged and can be added as fresh zones.
2) Paths that overlap with a single existing zone follow it consistently, thereby inheriting its color and maintaining feasibility.
3) Paths that would otherwise overlap with multiple zones are automatically redirected at their first intersection, because BFS continues only along the first touched color. This ensures loop-free and overlap-consistent integration without post-processing.

Thus, the BFS not only identifies augmenting paths but also enforces the overlap rules by construction.

B. Online Integration Routine

Receivers are processed sequentially, one at a time. For a given receiver r_i, the constrained BFS is invoked repeatedly until no more admissible augmenting paths can be found. Each discovered path is backtracked, assigned a color, and appended to the multicast edge set E_M. The update rules are as follows:

1) **No overlap (disjoint path)**: a fresh color is drawn from the palette and assigned to all edges in the path. The path forms a new independent zone.
2) **Single overlap (aligned path)**: the path inherits the color of the zone it touches, and its edges are appended to E_M under that color.
3) **Redirected multi-overlap**: if the BFS encounters multiple zones, it automatically continues along the first touched zone. The resulting path is reconstructed as the prefix up to the intersection plus the existing suffix of the touched zone. This guarantees loop-free redirection and consistent color assignment.

After integration, residual capacities are updated by saturating the edges of the discovered path, ensuring that no edge instance is reused beyond its unit capacity. The routine then attempts another BFS for the same receiver until its maximum flow is exhausted, after which the algorithm proceeds to the next receiver. In this way, the multicast subgraph is constructed online, with overlap control embedded in every step.

The complete procedure is summarized in Algorithm 2. The pseudocode integrates path discovery, color assignment, residual updates, and receiver-by-receiver processing into a single unified routine. It makes explicit the initialization of variables, the BFS state expansion, the backtracking of paths, and the incremental updates to both the residual graph and the multicast subgraph. This detailed step-by-step description facilitates both theoretical analysis and practical implementation of the proposed framework.

C. Algorithm Walkthrough

For clarity, we now describe the main components of Algorithm 2 step by step, grouped by line ranges.

Lines 1–4 (Initialization). The algorithm begins by initializing the multicast edge set E_M and the residual edge set E_{res}, which contains all directed unit-capacity edges that remain unsaturated. A global color palette $\mathcal{C} = \{c_1, c_2, \dots\}$ is defined together with a pointer ix to allocate fresh colors. The function $\text{color}(\cdot)$ maps each edge to its assigned color, and initially all edges are uncolored.

Lines 5–6 (Receiver loop). Receivers are processed sequentially. For each receiver r_i, we maintain three sets: $\mathcal{P}_i$ (the set of paths integrated for r_i), C_i (the colors already used by r_i to ensure one color per path), and $\mathcal{E}_i$ (the edges already reserved for r_i to guarantee edge-disjointness of its paths). The outer while-loop attempts to find multiple admissible $s \to r_i$ paths until none remain.

Lines 7–13 (BFS setup). A new breadth-first search (BFS) is initialized from $(s, \varnothing)$, where the state includes both the current node and the active color. The predecessor maps $\pi(\cdot,\cdot)$ and $\pi_{\text{edge}}(\cdot,\cdot)$ are cleared, and the source is marked as discovered. The variable *prevPaths* records the number of

Algorithm 2 Online Color-Constrained Multicast Construction

Require: $G = (V, E)$; source s; receivers $\{r_1, \dots, r_m\}$
Ensure: Multicast subgraph $G_M = (V, E_M)$ and $\{\mathcal{P}_i\}_{i=1}^{m}$

$E_M \leftarrow \varnothing$; $E_{\text{res}} \leftarrow E$ ▷ Residual Edges
$\mathcal{C} \leftarrow \{c_1, c_2, \dots\}$; $ix \leftarrow 0$ ▷ Color palette and pointer
color : $E \to \mathcal{C} \cup \{\varnothing\}$ ▷ Color function
color$(e) \leftarrow \varnothing,\ \forall e \in E$ ▷ All edges start uncolored
for $i = 1$ **to** m **do** ▷ Process receivers r_i
  $\mathcal{P}_i \leftarrow \varnothing$; $C_i \leftarrow \varnothing$; $\mathcal{E}_i \leftarrow \varnothing$ ▷ Paths, colors, edges
  while true do
    prevPaths $\leftarrow |\mathcal{P}_i|$
    $Q \leftarrow [\,(s, \varnothing)\,]$ ▷ BFS queue (node, active-color)
    $\pi(v, c) \leftarrow \varnothing,\ \forall v \in V,\ \forall c \in \mathcal{C} \cup \{\varnothing\}$
    $\pi_{\text{edge}}(v, c) \leftarrow \varnothing,\ \forall v \in V,\ \forall c \in \mathcal{C} \cup \{\varnothing\}$
    $\pi(s, \varnothing) \leftarrow (s, \varnothing)$ ▷ Source discovered
    while $Q \neq [\,]$ **do**
      $(u, c^\star) \leftarrow \text{head}(Q)$; $Q \leftarrow \text{tail}(Q)$
      for each $e = (u, v) \in E_{\text{res}}$ **with** $c_f(e) = 1$ **and** $e \notin \mathcal{E}_i$ **do**
        $c_e \leftarrow \text{color}(e)$ ▷ $c_e \in \mathcal{C} \cup \{\varnothing\}$
        if $c_e \in C_i$ **then continue**
        end if ▷ Already used color
        if $c^\star = \varnothing$ **then**
          $c_{\text{next}} \leftarrow c_e$ ▷ Uncolored
        else
          if $c_e \neq \varnothing$ **and** $c_e \neq c^\star$ **then continue**
          end if ▷ Forbid multi-color overlap
          $c_{\text{next}} \leftarrow c^\star$
        end if
        if $\pi(v, c_{\text{next}}) = \varnothing$ **then**
          $\pi(v, c_{\text{next}}) \leftarrow (u, c^\star)$; $\pi_{\text{edge}}(v, c_{\text{next}}) \leftarrow e$
          if $v = r_i$ **then** ▷ Receiver reached
            // Backtrack and update E_{res}
            $p \leftarrow [\,]$; $(x, c_x) \leftarrow (r_i, c_{\text{next}})$
            while $x \neq s$ **do**
              $e_x \leftarrow \pi_{\text{edge}}(x, c_x)$; $p \leftarrow [\,e_x\,] \,\|\, p$
              $(x, c_x) \leftarrow \pi(x, c_x)$
            end while
            $\Omega(p) \leftarrow \{\,\text{color}(e) : e \in p,\ \text{color}(e) \neq \varnothing\,\}$
            if $|\Omega(p)| = 0$ **then** ▷ No overlap
              $ix \leftarrow ix + 1$; $c \leftarrow c_{ix}$
            else
              $c \leftarrow \text{Only}(\Omega(p))$
            end if
            color$(e) \leftarrow c, \forall e \in p$
            $E_M \leftarrow E_M \cup p$; $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{p\}$
            $C_i \leftarrow C_i \cup \{c\}$; $\mathcal{E}_i \leftarrow \mathcal{E}_i \cup p$
            for each $e \in p$ **do** $c_f(e) \leftarrow 0$
            end for ▷ Residual update
            break ▷ Path found
          else $Q \leftarrow Q \,\|\, [\,(v, c_{\text{next}})\,]$
          end if
        end if
      end for
      if $|\mathcal{P}_i| >$ *prevPaths* **then break**
      end if
    end while
    if $|\mathcal{P}_i| =$ *prevPaths* **then break** ▷ BFS exhausted
    end if
  end while
end for
return $G_M = (V, E_M)$ and $\{\mathcal{P}_i\}_{i=1}^{m}$

paths found so far for r_i and will be used to determine whether the BFS successfully augmented the flow.

Lines 14–29 (BFS exploration). Nodes are extracted from the queue one by one. For each outgoing residual edge $e = (u, v)$ with capacity one and not already used by r_i, the algorithm evaluates its color c_e. Two constraints are enforced: (i) a receiver cannot reuse a color it has already consumed across its paths ($c_e \notin C_i$), and (ii) if the path is already following a color $c^\star$, then only edges that are uncolored or matching $c^\star$ may be traversed (multi-color overlap is forbidden). The resulting color state c_{next} is attached to the successor state (v, c_{next}). If this state has not been visited, it is discovered and linked through the predecessor maps.

Lines 30–49 (Path reconstruction and integration). If the sink r_i is reached, the shortest admissible augmenting path is reconstructed by backtracking predecessors from (r_i, c_{next}) to $(s, \varnothing)$. The set $\Omega(p)$ collects the colors already present on the path:

- If $\Omega(p) = \varnothing$, the path introduces no overlap, and a fresh color c_{ix+1} is assigned.
- Otherwise, by construction, $\Omega(p)$ contains exactly one color, which is reused consistently for the entire path.

All edges of p are assigned the selected color c, added to E_M, and appended to $\mathcal{P}_i$. The color c and the edges of p are also recorded in C_i and $\mathcal{E}_i$. Finally, the residual graph is updated by saturating each used edge ($c_f(e) \leftarrow 0$), ensuring that the same edge cannot be reused by another path. After this, the BFS terminates and the outer while-loop resumes to attempt another augmentation for r_i.

Lines 50–56 (Termination for receiver r_i). If no new path was discovered in the last BFS run ($|\mathcal{P}_i| =$ *prevPaths*), the while-loop ends, meaning the maximum flow to r_i under the color constraints has been reached. The algorithm then proceeds to the next receiver.

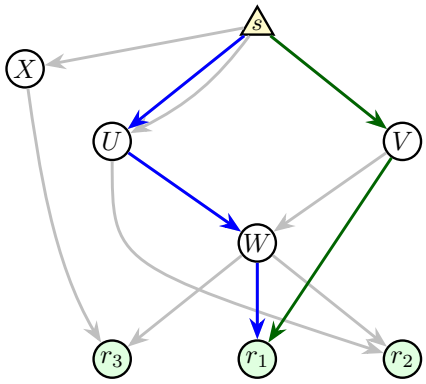
Final return. After all receivers have been processed, the algorithm outputs the multicast subgraph $G_M = (V, E_M)$ together with the per-receiver path sets $\{\mathcal{P}_i\}_{i=1}^{m}$.

D. Illustrative Example

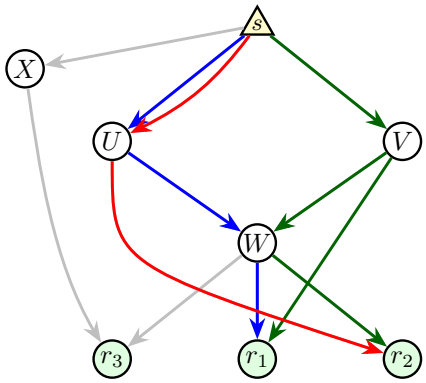
To illustrate the operation of the proposed online algorithm, we now present a step-by-step example on an illustrative topology. The goal is not to represent a large-scale network, but rather to make visible how the color-constrained BFS and the online integration rules interact when multiple receivers are sequentially added. For clarity, all links in the figures are drawn in gray, and the active augmenting paths discovered by the algorithm are highlighted in color. Different colors represent distinct flow zones created during the process.

1) First Receiver: We begin with receiver r_1, assumed to have a min-cut of two with respect to the source s. The first BFS run discovers a path $s \to U \to W \to r_1$, which is disjoint and therefore assigned a fresh color (blue). A second BFS identifies an edge-disjoint path $s \to V \to r_1$, assigned another fresh color (green). At this point, the multicast subgraph E_M for r_1 consists of two independent zones, guaranteeing that r_1 receives two linearly independent substreams. Figure 3 (a) shows this stage.

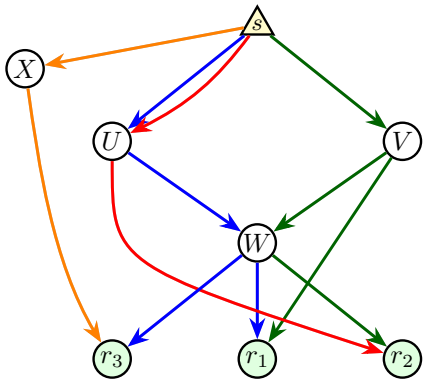
2) Second Receiver: Next, receiver r_2 is integrated. The first BFS discovers a path $s \to U \to r_2$, which is completely disjoint from the existing zones; it is therefore assigned a fresh color (red). A second BFS then identifies a path $s \to V \to W \to r_2$, which touches the preexisting green zone at edge (s, V) and consistently follows it. This path is thus integrated into the green zone. As a result, r_2 receives two independent substreams, one red and one green. Figure 3 (b) illustrates this integration.



(a) First receiver r_1: two edge-disjoint paths.



(b) Second receiver r_2: one disjoint path and one single-color overlap.



(c) Third receiver r_3: two paths: one redirected and one disjoint.

Fig. 3: Step-by-step online integration of three receivers: (a) disjoint paths for r_1 (blue, green); (b) integration of r_2 with one disjoint path (red) and one single overlap (green); (c) integration of r_3 with redirection at the first overlap (blue).

3) Third Receiver: Finally, receiver r_3, also with min-cut two, is processed. The first BFS identifies a disjoint path $s \to X \to r_3$, which does not intersect any existing zone. This creates a fresh zone, highlighted in orange. A second BFS then finds a candidate path through W that would intersect multiple zones. Instead of allowing a multi-overlap, the algorithm redirects the suffix at the first touched zone (blue), forcing the path to align entirely with it. The resulting path therefore consists of a blue-colored suffix from $W \to r_3$, ensuring feasibility and preventing cycles. At this stage, r_3 receives two substreams: one disjoint (orange) and one aligned with an existing zone (blue). Figure 3 (c) illustrates this final integration.

E. Discussion

These examples highlight how the online algorithm adapts naturally to successive receiver integrations. Each new receiver is guaranteed to receive the maximum number of admissible substreams permitted by the network min-cut, while overlaps are managed in a way that is consistent with source-only coding. The use of colors makes the invariants transparent: each receiver obtains substreams of distinct colors, no path uses edges of more than one color, and redirections occur automatically at the first overlap.

This illustrative topology also emphasizes the scalability advantage of the online approach: each integration is performed incrementally without recomputing global flows. In the next section, we formalize this intuition by analyzing the algorithm's complexity and comparing it with classical Edmonds–Karp and coding-optimal benchmarks.

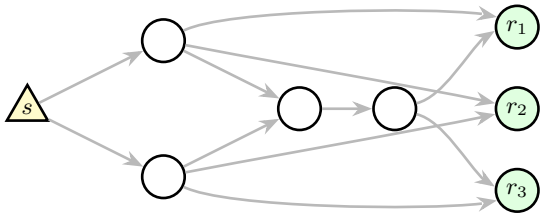
VI. Theoretical Analysis

A. Correctness

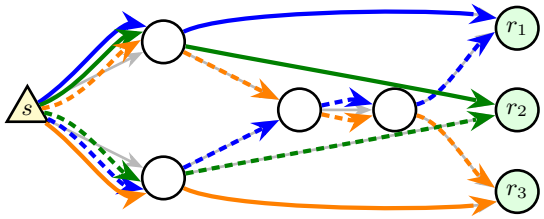
The proposed algorithm admits a theoretical characterization in terms of correctness, feasibility, and its relation to classical notions of maximum flow and multicast construction. The following discussion highlights its fundamental properties.

The algorithm guarantees loop-free integration of receiver flows. By construction, every newly added path either introduces a disjoint flow or aligns with a single existing flow upon its first overlap. As a result, each directed edge instance is saturated at most once, ensuring strict adherence to unit-capacity feasibility. Furthermore, the breadth-first nature of the search ensures that augmenting paths are the shortest admissible ones in hop count, consistent with the classical Edmonds–Karp framework.

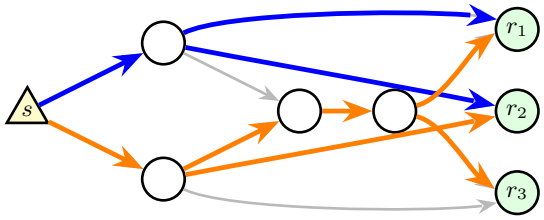
From the perspective of throughput, the online construction is always bounded above by the maximum-flow value $f(s, r_i)$ for each receiver. In the absence of multi-path overlaps, this bound is achieved exactly. When redirection occurs, the resulting path set may fall short of $|f(s, r_i)|$ disjoint paths, but the loss is limited in practice. In our ER/WS experiments we did not observe a degradation of the group multicast max-flow when comparing unconstrained EK with its restricted variant (cf. Section VIII). Nonetheless, this near-optimality is not universal: there exist small, carefully crafted unit-capacity digraphs where the overlap constraint strictly reduces the achievable multicast throughput. Figure 4 provides a minimal counterexample: per-receiver EK finds two edge-disjoint paths to every receiver, whereas the restricted integration forces a first-touch alignment that prevents two independent zones from serving all receivers simultaneously, thereby reducing the multicast rate.



a) Topology before algorithms.



b) Per-receiver EK: each r_i attains two edge-disjoint paths.



c) Restricted EK (zones): r_1, r_2 use blue and orange; r_3 only orange.

Fig. 4: Adversarial topology showing a strict loss under overlap-constrained integration. (a) Topology only. (b) Per-receiver EK: two edge-disjoint paths to every receiver. (c) Restricted EK: the first-touch overlap rule collapses one branch for r_3, reducing the multicast group throughput.

A key structural property of the algorithm is its treatment of overlaps. By enforcing that overlaps are confined to a single preexisting path, the resulting multicast subgraph is naturally partitioned into edge-disjoint zones. This structure has two important implications. First, it ensures compatibility with source coding: each zone can be associated with a distinct substream, enabling receivers to reconstruct the transmitted content without requiring intermediate coding operations. Second, it yields a compact, tree-like topology: although the subgraph is not a tree in the strict sense, redirected flows converge consistently toward the source, producing a structure with branching behavior analogous to multicast trees.

When compared to alternative multicast approaches, the proposed framework occupies an intermediate position. Relative to tree-based multicast, it achieves higher resilience by preserving multiple flow diversity paths where available. Relative to fully coded multicast, it sacrifices only marginal throughput while avoiding the operational complexity of in-network coding. Thus, the algorithm combines the feasibility and efficiency of Edmonds–Karp with structural guarantees that make it source-coding compatible and practically deployable at scale. The previous analysis demonstrated that the proposed online construction yields a loop-free, capacity-feasible multicast subgraph that approaches the maximum attainable group flow. The next step is to specify how the source uses this structure to assign information to its outgoing links, thereby realizing source-only coding across the colored multicast tree.

B. Complexity Analysis

The online multicast algorithm inherits its fundamental complexity from Edmonds–Karp, while introducing lightweight color checks that do not alter the asymptotic order. Let $|E|$ denote the number of directed edge instances in the multigraph (counting parallels). For a fixed receiver r_i, each constrained breadth-first search (BFS) runs in $O(|E|)$ time. Since all edges have unit capacity, every augmentation increases the flow by exactly one, and at most $O(|E|)$ augmentations can be performed. Hence, the complexity per receiver session is $O(|E|^2)$. Processing all m receivers sequentially yields an overall complexity of $O(m|E|^2)$. The additional color checks performed during BFS expansion are constant-time operations, so they do not modify this bound.

For fairness, these figures correspond exclusively to the construction of multicast topologies. Classical Edmonds–Karp applied independently to each receiver without overlap restrictions has the same $O(m|E|^2)$ cost, but requires additional reconciliation of overlapping paths after the fact. Tree-based heuristics, such as shortest-path or Steiner approximations, typically range between $O(|E|\log|V|)$ and $O(|V||E|)$ in complexity, and are therefore computationally lighter, but structurally limited since they cannot exploit multiple edge-disjoint paths and thus fall short of the throughput achieved by flow-based methods.

In comparison, constructing network coding–optimal multicast subgraphs also requires computing maximum flows for each receiver, leading again to $O(m|E|^2)$ complexity at the topology stage. However, full network coding adds an additional layer of processing: encoding at the source and decoding at the receivers involve solving systems of linear equations whose dimension scales with the number of receivers and active subflows, resulting in at least $O(m^3)$ computational cost, not accounting for per-packet operations at intermediate nodes. By contrast, the source-coding paradigm enabled by our online algorithm requires only lightweight linear combinations at the source and no in-network coding, which drastically reduces implementation complexity while preserving near-optimal throughput.

VII. Source-Only Information Encoding

The online algorithm described in Sections IV–VI constructs a multicast subgraph $G_M = (V, E_M)$ that satisfies the structural constraints required for source-only coding: each receiver r_i is connected to the source s through k_i edge-disjoint or zone-aligned paths, and every path of a receiver is assigned a distinct color. This Section completes the framework by defining the source-side rule that determines what information

is transmitted on each outgoing edge of s, thus achieving the multicast group maximum flow without in-network recoding.

Once the online algorithm has produced the colored multicast subgraph, each color identifies a distinct transmission branch originating at the source. Different colors may reach disjoint or overlapping subsets of receivers, and the total number of colors $|\mathcal{Z}|$ is generally greater than or equal to the multicast max-flow value. Hence, the colored structure defines the potential information outlets from the source, but not yet the specific symbols to be transmitted on them. At this stage, the symbol-assignment procedure proposed by [13], [14] is applied to determine, for each color, whether the transmitted content should be a clear (uncoded) symbol or a linear combination of source symbols, depending on the subset of receivers reached by that color. For instance, if the maximum multicast flow is $K = 1$ and two colors depart from the source to reach disjoint receiver subsets, both colors carry the same clear symbol, with no coding required. In contrast, when receivers share parts of the multicast tree, the source assigns coded symbols across the corresponding colors to preserve the linear independence of the information observed at each receiver.

Following the formulation adopted throughout this paper, the multicast group throughput is limited by the receiver with the smallest individual max-flow:

$$K = \min_i f(s, r_i) = \min_i k_i, \tag{2}$$

so that all receivers can decode the same number of independent symbols. At each time slot, the source generates K base symbols $X = [x_1, x_2, \ldots, x_K]^T$, where each x_j belongs to a finite field $\mathbb{F}_q$. Each outgoing edge $e = (s, v)$ is associated with a color $z(e) \in \mathcal{Z}$, and the symbol transmitted along it is a linear combination

$$y_e = \mathbf{g}_e^T X, \tag{3}$$

where $\mathbf{g}_e \in \mathbb{F}_q^K$ is a coding vector drawn from a global source matrix G. The set of vectors $\{\mathbf{g}_e : e \in E_M,\ s \in e\}$ is chosen to be linearly independent, and all edges sharing the same color transmit identical mixtures, i.e. $y_e = y_{e'}$ whenever $z(e) = z(e')$. Hence, the color-to-vector mapping $\{z \mapsto \mathbf{g}_z\}$ is defined once at the source and reused consistently along the entire zone.

Each receiver r_i observes the symbols corresponding to the set of colors $Z_i \subseteq \mathcal{Z}$ that reach it, forming $Y_i = G_{Z_i,:} X$. Receiver r_i successfully decodes the K source symbols if the submatrix $G_{Z_i,:}$ has full rank K, i.e.

$$\operatorname{rank}\big(G_{Z_i,:}\big) = K. \tag{4}$$

This condition is guaranteed if G is constructed as a Vandermonde or Cauchy matrix over a field of size $q \geq |\mathcal{Z}|$, since any subset of at most K rows is linearly independent. Consequently, all receivers obtain $K = \min_i k_i$ independent combinations, matching the multicast group max-flow defined in (2).

To illustrate, consider a source connected to three receivers through three outgoing zones. With $K = 2$ and base symbols A and B, the source transmits $y_{z_1} = A$, $y_{z_2} = B$, and $y_{z_3} = A{+}B$. Receiver r_1 receives $\{A, B\}$, r_2 receives $\{A, A{+}B\}$, and r_3 receives $\{B, A{+}B\}$. Each receiver obtains two linearly independent combinations of $\{A, B\}$, thus recovering all symbols and achieving the group throughput $K = 2$. This example reproduces the principle proposed by [13], [14], generalized here to the online multicast topology produced by our framework.

A practical realization of the matrix G can be obtained using Reed–Solomon (RS) encoding over $\mathbb{F}_q$. In this case, the K source symbols correspond to the information symbols of an (N, K) MDS block code, and the set of $N = |\mathcal{Z}|$ c coded symbols $\{y_z\}$ is produced by evaluating the associated RS generator polynomial at N distinct field elements $\{\alpha_z\}$. This construction ensures that any subset of K coded symbols is linearly independent, thereby satisfying condition (4) by design. In practice, the source assigns one coded symbol y_z to each color and injects it into the network according to the mapping established by the online algorithm. Since intermediate nodes merely forward packets, the resulting scheme realizes source-only linear coding with the same algebraic guarantees as the Reed–Solomon code, while remaining fully compatible with the topology produced by the color-constrained construction.

In summary, once the multicast subgraph satisfies the single-overlap invariant, the source can directly map each color to a predetermined coding vector and transmit the corresponding linear combinations according to (3). Intermediate nodes simply forward packets of their assigned color, with no recoding or coordination required. Together with the online construction algorithm, this source-side information assignment provides a complete, constructive realization of the source-coded multicast paradigm, achieving the maximum feasible multicast throughput with minimal operational complexity.

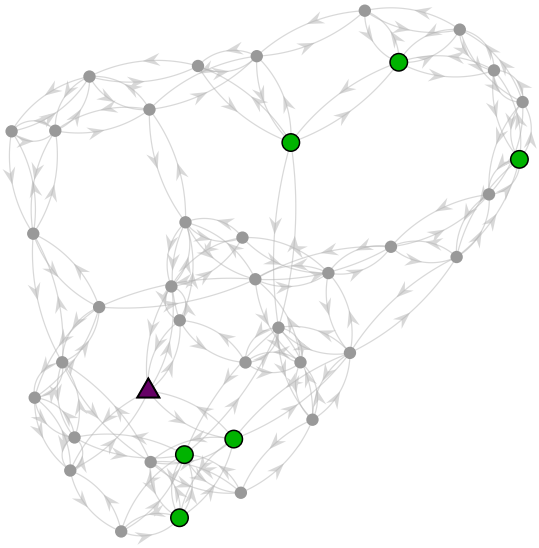
VIII. Performance Evaluation

A. Simulation Setup

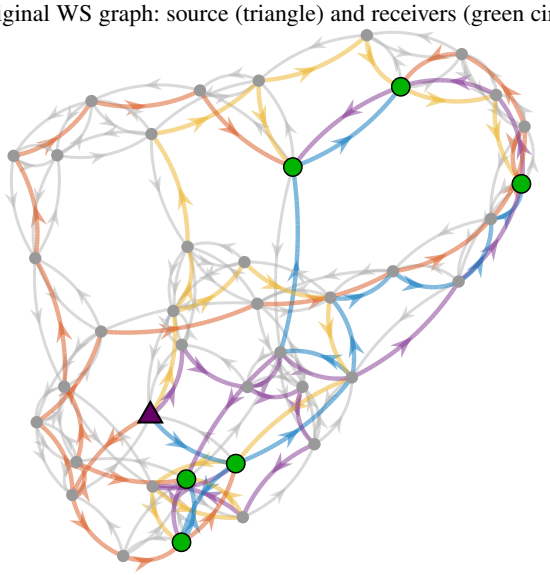
To evaluate the performance of the proposed framework, we carried out simulations on two widely used random graph models: the Erdős–Rényi (ER) model [23] and the Watts–Strogatz (WS) small-world model [24]. These models have been extensively employed in the networking literature to capture different structural properties: ER graphs provide a baseline of uniform randomness, while WS graphs incorporate both clustering and short average path length, thus better reflecting realistic communication networks.

For both ER and WS models, the total number of nodes was varied from 10 to 200 in increments of 5. The receiver set was chosen randomly with densities between 5% and 25%, in steps of 5%. Likewise, the link density was explored in the range 10% to 50%, also in steps of 5%. All edges were assigned unit capacity, and for each parameter setting multiple random graph realizations were generated; results were averaged to ensure statistical significance. For the WS model, the rewiring probability was fixed at $\beta = 0.1$, while in the ER model no such parameter is defined. In both cases, the expected mean degree is consistent with the chosen link density.

To build intuition, Figs. 5 and 6 show representative instances with 40 nodes, 6 receivers, and mean degree $k = 4$.

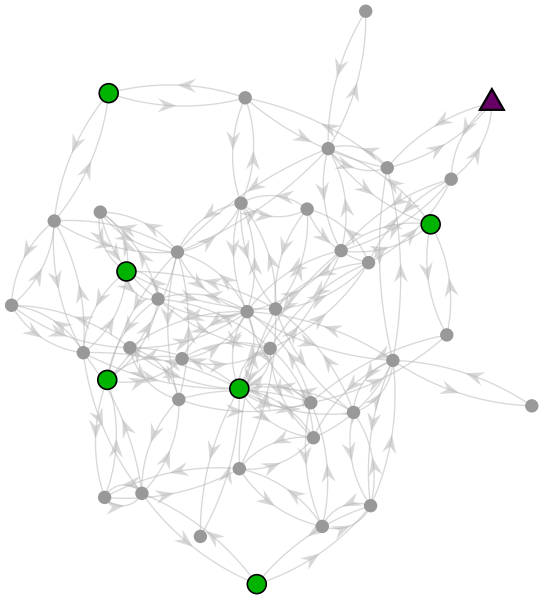


(a) Original WS graph: source (triangle) and receivers (green circles).

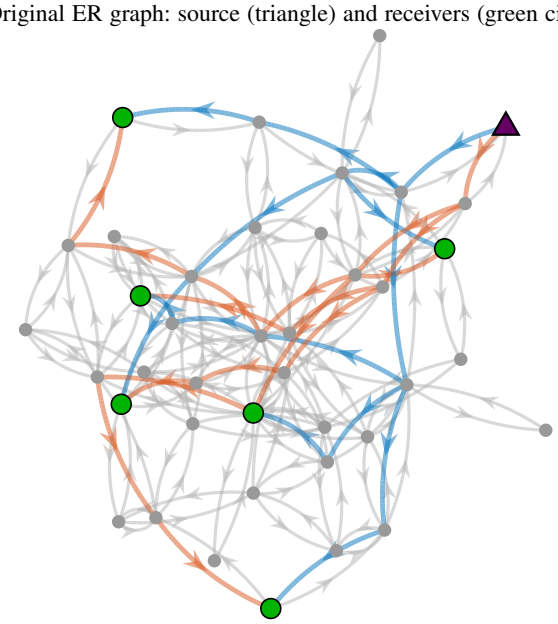


(b) Online multicast construction for WS.

Fig. 5: Illustrative Watts–Strogatz example (n=40, k=4, β=0.1).



(a) Original ER graph: source (triangle) and receivers (green circles).



(b) Online multicast construction for ER.

Fig. 6: Illustrative Erdős–Rényi example (n=40, p≈0.103).

The WS example (Fig. 5) corresponds to $\beta = 0.1$, while the ER example (Fig. 6) achieves the same expected degree with $p \approx 0.103$ (equivalently $m = 80$ edges). In both cases, panel (a) shows the original topology—with the source marked as a triangle and the receivers highlighted in green—and panel (b) depicts the corresponding online construction of the multicast subgraph. Together, these examples illustrate how both graph models can be instantiated under equivalent parameters, enabling a fair comparison of the proposed framework.

Finally, it is worth noting that, although the multicast max-flow towards the group remains unchanged, the restricted Edmonds–Karp variant may reduce the maximum flow observed at individual receivers.

B. Throughput Results

Across our main grid of ER and WS configurations, the online construction delivers identical multicast throughput to the offline benchmark obtained by running Edmonds–Karp independently for each receiver. This holds throughout the ranges considered for network size, edge density, and receiver density. Figures 7 and 8 report the aggregate multicast throughput as the receiver density ρ increases, confirming the coincidence between the restricted online construction and the unconstrained EK baseline across both random graph models.

A closer look at these figures reveals consistent scaling patterns across both topologies. For a fixed link density, the total multicast throughput increases with the network size n. This trend is expected since, under constant edge density, the

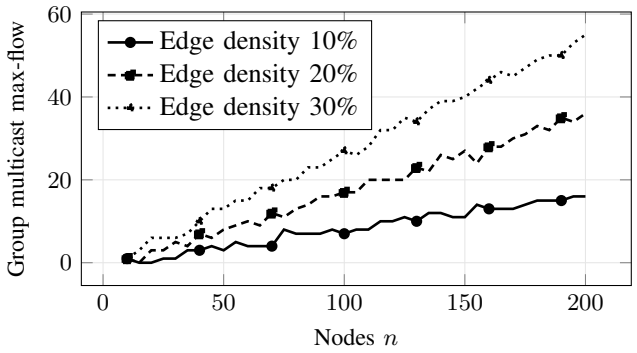


Fig. 7: WS ($\beta = 0.1$), receiver density 25%. Group multicast max-flow vs. n for **edge densities** 10%, 20%, and 30%. Restricted EK matches EK (curves overlap).

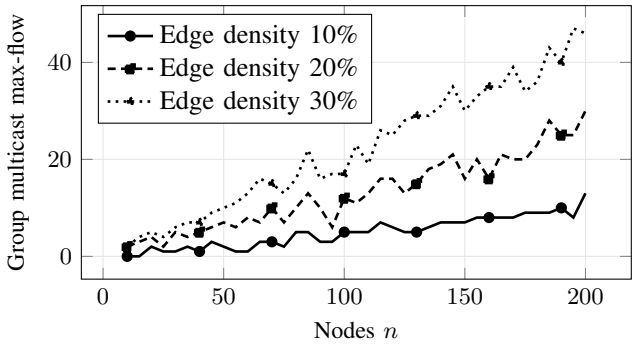


Fig. 8: ER, receiver density 25%. Group multicast max-flow vs. n for **edge densities** 10%, 20%, and 30%. Online matches EK (curves overlap).

absolute number of links grows quadratically with n, providing more potential disjoint routes between the source and the receivers. Similarly, higher link densities yield proportionally larger throughput values across all receiver densities, reflecting the linear dependence between aggregate capacity and connectivity.

When comparing the two models, WS graphs exhibit slightly steeper throughput curves than ER for equivalent link and receiver densities. This difference arises from the small-world structure of the WS model: its higher clustering and shorter characteristic path length enhance the availability of alternative low-hop routes, thus increasing the effective diversity of disjoint paths. In contrast, the ER model — lacking such structural shortcuts— shows smoother growth with n and lower marginal gains as receiver density increases. Nonetheless, both models confirm the same qualitative trends and, most importantly, the same quantitative match between the restricted and unconstrained Edmonds–Karp outcomes across the examined parameter space.

To further investigate the limits of this correspondence, we conducted a set of stress tests designed to exacerbate potential overlap conflicts in the online construction. Specifically, we fixed the mean degree to $k = 4$ in the WS model (with $\beta = 0.1$) and progressively scaled the network size n while imposing a dense receiver configuration ($r/n = 30\%$). Holding k constant isolates the effect of size under sparse

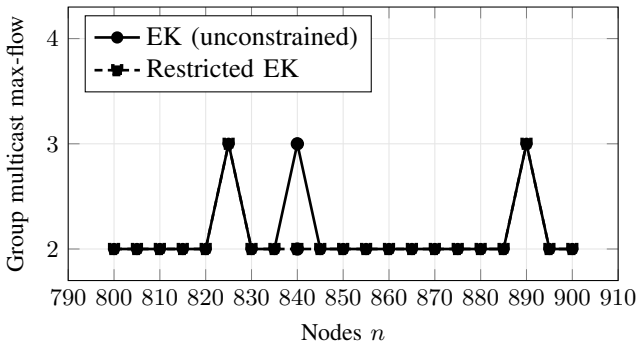


Fig. 9: WS (k=4, β=0.1). Group multicast max-flow vs. n, *zoom* a $n \in [800, 900]$. The $n = 840$ case shows a difference between both algorithms.

per-node capacity: as n increases, the number of directed edges grows linearly ($m = nk$), while the edge density $\rho = m/m_{\max} \approx k/(n-1)$ decays inversely with n. In parallel, the receiver set expands proportionally, amplifying fan-out pressure near the source and increasing the likelihood of path overlap. This regime is therefore the most adversarial for the restricted integration rule. As shown in Fig. 9, the restricted construction continues to match the unconstrained EK across almost all sizes; the *first* deviation appears at $n = 840$, with a maximum gap of only one flow unit thereafter.

These stress-test observations reinforce the theoretical expectations: although adversarial configurations can enforce a strict loss under the overlap constraint, such structures are extremely rare within ER or WS ensembles. Empirically, even under dense receiver sets and sparse degree, throughput losses remain infrequent, small in magnitude, and bounded by a single flow unit. This outcome demonstrates that the proposed online integration rule preserves near-optimal throughput in practically all random topologies, validating its efficiency and robustness as a scalable multicast construction method. After confirming that the proposed online framework matches the optimal throughput of the Edmonds–Karp benchmark, we now turn to its computational aspects. The following analysis examines how runtime scales with the number of nodes, link density, and receiver population in both ER and WS networks.

C. Scalability

We evaluate scalability by measuring runtime as a function of network size, receiver density, and average degree for both ER and WS ensembles. Results consistently show that execution time grows approximately quadratically with the number of edges, in line with the expected $O(|E|^2)$ bound per receiver. However, the proposed online construction achieves substantial practical gains by eliminating redundant recomputations and incrementally reusing previously integrated flow structure.

Across all tested configurations, the online algorithm maintains the same multicast throughput as the reference Edmonds–Karp baseline while progressively reducing runtime as network connectivity increases. Higher link densities, which correspond to larger average node degree, yield proportionally shorter

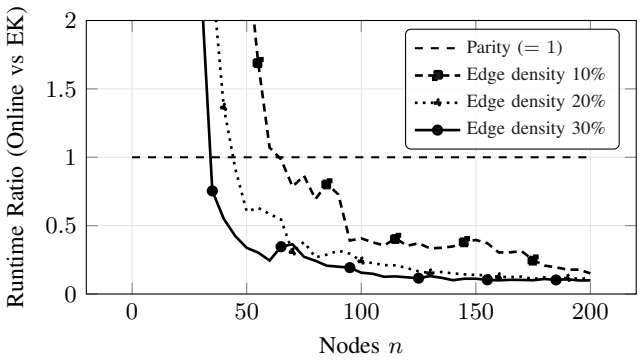


Fig. 10: Runtime ratio of the online construction relative to the per-receiver Edmonds–Karp baseline for Watts–Strogatz graphs ($\beta = 0.1$) with a receiver density of 25%. The curves correspond to edge densities of 10%, 20%, and 30%. The dashed line indicates parity ($= 1$); values below one denote faster execution of the online algorithm.

execution times relative to EK. This effect stems from the online method's ability to reuse colored paths and merge zones efficiently: denser topologies provide more alternative disjoint routes, lowering the number of residual searches and augmentations required per receiver.

Figure 10 illustrates this behavior for the Watts–Strogatz (WS) model with rewiring probability $\beta = 0.1$. The plot reports the runtime ratio between the proposed online algorithm and the per-receiver Edmonds–Karp baseline as the number of nodes n increases. The dashed line marks parity ($= 1$), and values below it indicate faster execution of the online algorithm. The ratio decreases monotonically with link density, confirming that efficiency improves as connectivity grows. Equivalent trends were observed for Erdős–Rényi (ER) graphs under the same parameter settings, indicating that the observed scalability improvement is consistent across both random graph models.

The improvement is particularly pronounced in small-world structures, where high clustering and short characteristic path lengths favor path reuse and further reduce exploration overhead. Overall, empirical results confirm that the online construction scales gracefully with network size and density: it preserves optimal multicast throughput while achieving significant and increasing computational efficiency as the underlying connectivity grows.

IX. Conclusion and Future Work

This paper introduced an online source-coded multicast framework that incrementally integrates receiver flows while explicitly controlling path overlaps. By embedding color-constrained breadth-first search into the path discovery process, the algorithm guarantees that each new flow is either disjoint, aligned with a single preexisting flow, or redirected at its first intersection. This construction preserves unit-capacity feasibility, enforces overlap consistency, and ensures compatibility with source-only coding —all without requiring global recomputation.

Our theoretical analysis established that the algorithm operates in $O(|E|^2)$ time per receiver, on par with classical Edmonds–Karp, while enforcing structural invariants absent in offline approaches. Extensive simulations on Erdős–Rényi and Watts–Strogatz random networks demonstrated that, in practice, the online construction consistently matches the maximum-flow throughput benchmark, confirming its robustness and efficiency. Moreover, the resulting multicast subgraphs offer enhanced resilience and path diversity compared to tree-based methods, while avoiding the complexity of intermediate-node coding inherent to network coding.

Future work. Several directions remain open. First, while no throughput degradation was observed in random networks, structured topologies such as scale-free (Barabási–Albert) or stochastic block models may reveal conditions where the online constraints limit achievable flow. Second, further analysis of resilience under dynamic conditions (e.g., receiver churn, link failures) would strengthen the practical relevance of the approach. Third, integrating lightweight coding extensions at the source (e.g., adaptive redundancy or hybrid substream designs) may improve robustness to packet loss while preserving the no-recoding property at intermediate nodes. Finally, applying the framework to real-world network traces and deployments would provide additional validation of its scalability and implementability. Overall, the proposed online algorithm provides a practical middle ground between tree-based multicast and full network coding, offering a scalable, source-only solution for next-generation group communication services.

References

[1] J. Moy, "Multicast routing in datagram internetworks and extended lans," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 55–64, 1994.

[2] J. Widmer and J. L. Boudec, "Extending a multicast tree using diversity-based routing," in *Proceedings IEEE INFOCOM 2001*, vol. 2. IEEE, 2001, pp. 924–933.

[3] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1. ACM, 2002, pp. 1–12.

[4] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian J. of Mathematics*, vol. 8, pp. 399–404, 1956.

[5] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," *Soviet Mathematics Doklady*, vol. 11, pp. 1277–1280, 1970.

[6] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[7] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[8] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. on Inf. Theory*, vol. 49, no. 2, pp. 371–381, feb 2003.

[9] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, oct 2003.

[10] P. A. Chou, Y.-H. Wu, and K. Jain, "Practical network coding," in *Annual Allerton Conference on Communication, Control, and Computing*, 2003, pp. 40–49.

[11] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," in *IEEE Transactions on Information Theory*, vol. 52, no. 10. IEEE, 2006, pp. 4413–4430.

[12] T. Lestayo, V. M. Fern'andez, and C. López, "Adaptive approach for fec reliable multicast," *Electronics Letters*, vol. 37, no. 22, pp. 1333–1335, Oct. 2001.

[13] T. Lestayo-Martínez and M. Fernández-Veiga, "Source-coded multicast for efficient content delivery," *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 140–145, 2023.

[14] T. Lestayo and M. Fernández, "Source-coded multicast with single and aggregated sources for efficient content delivery," *IEEE Open Journal of the Communications Society*, 2024.

[15] D.-Z. Du and F. K. Hwang, "Approximation algorithms for the steiner tree problem in networks," in *Networks*, vol. 21. Wiley Online Library, 1991, pp. 73–82.

[16] F. K. Hwang and D. S. Richards, "Steiner tree problems," *Networks*, vol. 22, no. 1, pp. 55–89, Jan. 1992.

[17] M. Charikar, C. Chekuri, T.-y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed steiner problems," *Journal of Algorithms*, vol. 33, no. 1, pp. 73–91, Oct. 1999.

[18] G. N. Rouskas, "Multicast routing and wavelength assignment in optical networks," in *Proceedings of IEEE INFOCOM*. IEEE, 1997, pp. 188–195.

[19] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. on Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, jul 2000.

[20] D. Lun, N. Ratnakar, R. Koetter, and M. Médard, "Network coding for efficient wireless unicast," in *Proceedings of the 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005, pp. 621–630.

[21] V. N. Padmanabhan, L. Qiu, and G. M. Voelker, "Resilient peer-to-peer streaming," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003, pp. 73–82.

[22] S. Kandula, D. Katabi, B. Davie, and H. Chiu, "Walking the tightrope: responsive yet stable traffic engineering," in *ACM SIGCOMM 2005*. ACM, 2005, pp. 253–264.

[23] P. Erdős and A. Rényi, "On random graphs i," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

[24] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.