MULTI-GRAPH REGULARIZATION FOR EFFICIENT DELIVERY OF USER GENERATED CONTENT IN ONLINE SOCIAL NETWORKS

Jacob Chakareski
Signal Processing Laboratory - LTS4, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

ABSTRACT
We present a methodology for enhancing the delivery of user-generated content in online social networks. To this end, we first regularize the social graph via node capacity and link cost information associated with the underlying data network. We then design a technique for constructing the most efficient delivery tree over the regularized social graph. Finally, we derive an optimization algorithm for allocating the nodes' uplink capacities over the content distribution tree. Our system substantially outperforms the conventional method of flooding data over the social graph, over multiple criteria. In particular, a 100% reduction in terms of network cost and data delivery delay is registered.

1. INTRODUCTION
The technological advances in video compression and broadband communications have enabled a virtual revolution in online content sharing via social networks. For example, according to recent statistics [1] there are 24 hours of content uploaded on YouTube [2] every minute and the top ten user-run channels there consistently exceed an audience of one million subscribers. Furthermore, traffic monitoring studies predict an even bigger wave of such video content in the future that will flood our data networks and challenge their long-established founding principles [3].

At present, content sharing in social networks is typically performed via flooding. In particular, we virally send the content of interests to all our contacts (neighbours) in the social graph, who then repeat the same procedure with their own contacts, etc. Unfortunately, such a mode of content delivery is very inefficient from a data communication perspective and can lead to poor timeliness of the multimedia application comprising the content. The present paper provides a methodology that effectively overcomes these difficulties.

In our system, the social graph is regularized with data network information in order to enable efficient content sharing between its nodes. Specifically, we assign weights to the edges of the graph that take into account the nodes’ uplink capacities and the edges’ data transport costs. Then, we compute a maximum weight spanning tree over the regularized graph, starting from the source of the content as its root. Delivering data over such a tree will minimize the associated network cost. Finally, we optimally allocate the nodes’ uplink capacities across their children in the tree so that in addition the data delivery delay of the multicast tree is minimized. As our experimental results show, our framework significantly outperforms the conventional method of content flooding, both in terms of network efficiency and data latency.

Prior work related to the present paper includes [4] and [5] that focus on exploiting, in the context of peer-to-peer networks, the existence of social phenomena such as communities of users with similar interests, in order to provide enhanced content dissemination by social-based discovery and recommendation systems. Furthermore, in [6] a tree-based approach for user generated video dissemination in online social networks is described. The proposed system constructs an overlay tree for data transmission which takes advantage of download-and-play users to achieve better live streaming performance. Finally, the work in [7] takes into account the users’ content preferences and the physical links’ capacities and costs in order to improve the information flow-cost ratio of media delivery in online social networks.

2. SOCIAL GRAPH REGULARIZATION
Let $G = (V, E)$ denote the graph representing the online social network under consideration where $V$ and $E$ correspond respectively to the sets of vertices and edges of the graph. Each node $i \in V$ is associated with an upload bandwidth capacity $C_i$ characterizing its Internet access link, as provided by the node’s ISP. This is the maximal data rate at which $i$ can send data to other nodes in the graph. Furthermore, let $c_{ij}$ denote the network cost of sending data between nodes $i$ and $j$ in the social graph. In particular, this is the cost that the underlying transport network operator will experience when delivering a unit of data from node $i$ to node $j$.

We will use the information associated with the underlying data network to regularize the social graph in order to make it more efficient for sharing user generated content between its nodes. In particular, to each edge $e \in E$ directed from node $i$ to node $j$ in the social graph we assign a composite weight $w_e$ that is computed as the ratio of $C_i$ and $c_{ij}$. Through $w_e$ we simultaneously capture the influence of two factors affecting the process of delivering content over the social graph. Specifically, the nominator accounts for the serving capacity of node $i$, while the denominator will differentiate link $e$ relative to other edges in the graph according to the associated network cost of sending data over it.

3. TREE CONSTRUCTION
We deliver the content over a directed tree (arborescence) rooted at the content creator. The dissemination tree is computed over the regularized social graph such that its overall weight is maximized. This will enable serving the content efficiently and in an expedient manner, simultaneously. Constructing a maximum (or minimum) arborescence for a graph has been studied in the past in the context of computer networks [8, 9]. Our algorithm is inspired by the technique originally proposed in [9]. We briefly describe it here for completeness:

- Let $G^0 = G$, where $V^0 = V$ and $E^0 = E$. Let $j = 0$.
- Step 1. For every node $i \in V^j$ select an incoming edge with maximum weight. The set of these arcs is denoted $E^j$.
- Step 2. If there is no loop in $E^j$, then the process stops and $T = (V^j, E^j)$ is the maximum weight arborescence of the graph $G^j$. Go to Step 4. Otherwise,
- Step 3. Contract each cycle in $E^j$ into a single vertex. Let the resulting graph be denoted as $G^{j+1} = (V^{j+1}, E^{j+1})$. The weight
of each edge $e_j^{j+1} \in E_j^{j+1}$ is (re)defined using one of the following two cases

$$w_{e_j^{j+1}} = \begin{cases} w_{e_j} \\ w_{e_j} + w_{e_j^0} - w_{e_j} \end{cases},$$

(1)

$$w_{e_j^{j+1}} = \begin{cases} w_{e_j} \\ w_{e_j} + w_{e_j^0} - w_{e_j} \end{cases},$$

(2)

where (1) applies when $e_j^{j+1}$ is not incident to a vertex in $G_j^{j+1}$ representing a contracted cycle from $G_j$, while (2) applies when the converse is true. In this latter case, $e_j^0$ is the edge in the cycle with minimum weight and $e_j$ is the edge in the cycle incident to the same node as $e_j^0$ (all with reference to the graph $G_j$).

If $e_j$ is not incident to a vertex in $G_j^{j+1}$ representing a contracted cycle from $G_j$, then return to Step 1.

Step 4. Extend the arborescence $T_j$ into an arborescence $T_{j-1}$ of graph $G_{j-1}$. Specifically, let $e_j^1$ be a vertex in $T_j$ corresponding to a contracted cycle in $G_{j-1}$. Then,

(i) If $e_j^1$ is the root of $T_j$, select the minimum weight edge in the corresponding cycle. The remaining edges of the cycle together with the set $E_j^1$ form a subgraph of $G_{j+1}^{j+1}$ that represents the maximum weight arborescence $T_{j-1}$.

(ii) Otherwise, there is a unique edge $e_j^1 \in E_j^1$ incident to $e_j^1$. The corresponding edge $e_j^{j+1} \in E_j^{j+1}$ is incident to a node in the corresponding cycle in $G_{j-1}^{j-1}$. Remove the edge in the cycle that is incident to this same node. Again, the remaining edges of the cycle together with the set $E_j^1$ form the maximum weight arborescence $T_{j-1}$.

$j = j - 1$. Return to Step 4 until $j > 0.$

4. RATE ALLOCATION

Once the delivery tree is constructed, we are interested in assigning forwarding rates to each active link in the network such that the delay in serving a unit of data to every node in the tree is minimized. In addition to being cost efficient, this will also make the tree adept for delivering time-sensitive data, such as multimedia.

In particular, let $\mathcal{N}_i$ denote the set of children nodes for node $i$ in the tree. The serving delay per unit cost that node $j \in \mathcal{N}_i$ experiences can be written as a sum of two terms. The first term represents the corresponding delay of its parent $i$, while the second term corresponds to the latency in forwarding data from the link $i \rightarrow j$. That is

$$d_j = d_i + \frac{1}{r_{ij} - c_{ij}},$$

where $r_{ij} \leq C_1$ denotes the rate that node $i$ uses to forward data to node $j$. As stated above, we are interested in minimizing the overall delay of the delivery tree such that the uplink capacities of each node are not exceeded, i.e.,

$$\min \left\{ r_{ij} \right\}_{i \rightarrow j} \sum_{j \in \mathcal{N}_i} d_j \text{ s.t. } \sum_{j \in \mathcal{N}_i} r_{ij} \leq C_1, \forall i.$$  \hspace{1cm} (3)

It can be shown that the first sum in (3) can be written as $\sum_{i \rightarrow j} \frac{[T_j]}{r_{ij} - c_{ij}},$ where $T_j$ denotes the sub-tree rooted at node $j$. Therefore, the optimization in (3) can be decomposed as an optimization over every inner node in the tree, independently. Furthermore, using the method of Lagrange multipliers [10] the problem can be re-formulated as a non-constrained optimization that is written as follows

$$\min \left\{ r_{ij} \right\}_{j \in \mathcal{N}_i} \sum_{j \in \mathcal{N}_i} \frac{[T_j]}{r_{ij} - c_{ij}} + \lambda \left( \sum_{j \in \mathcal{N}_i} r_{ij} - C_1 \right), \forall i, \hspace{1cm} (4)$$

where $\lambda > 0$ is a Lagrange multiplier.

Finally, with some work a solution to (4) can be derived from its corresponding Karush-Kuhn-Tucker (KKT) conditions. Due to space constraints, we can only include here the final functional forms of the optimal forwarding rates $r_{ik}^*$, i.e.,

$$r_{ik}^* = \frac{C_i}{1 + \sum_{j \in \mathcal{N}_i, j \neq k} \sqrt{\frac{C_j}{C_j}} \cdot \frac{|T_j|}{|T_k|}},$$

(5)

$$r_{ij}^* = \frac{C_j}{1 + \sum_{k \in \mathcal{N}_j, k \neq i} \sqrt{\frac{C_i}{C_i}} \cdot \frac{|T_i|}{|T_k|}} \forall j \in \mathcal{N}_i, j \neq k.$$  \hspace{1cm} (6)

5. EXPERIMENTS

Here, we study through simulations the performance of our optimization techniques for enhancing the delivery of user generated content (UGC) in online social networks. Specifically, we first examine the efficiency of the tree construction algorithm from Section 3, as a function of the amount of transport network information taken into account for regularizing the social graph. Then, we compare the cost efficiency and delay performance of the proposed tree construction and rate allocation techniques against those of two reference schemes for delivering UGC in social networks.

In the first set of experiments, we employ a set of scale-free small-world random graph topologies that we generated synthetically such that they correspond to examples of online social networks published in the literature [11, 12]. The uplink capacities of the nodes in the social graph follow a distribution where 75% of them are characterized as having DSL/cable modem access link characteristics ($C_i = 300$ kbps), while the rest are modeled as Ethernet peers ($C_i = 750$ kbps). The network cost $c_{ij}$ of sending data from node $i$ to node $j$ in the graph, as introduced in Section 2, exhibits a uniform distribution over the range $[1, 16]$.

In these experiments, the performance of the tree construction algorithm when the complete transport network information is taken into account, as defined in Section 2, is denoted as $SN+NET$. On the other hand, the case when only the costs $c_{ij}$ of the network links $i \rightarrow j$ are considered is denoted as $LinkGain$ ($w_{ij} = 1/c_{ij}$). Similarly, $UplinkCapac$ denotes the case when only the uplink capacities of the nodes are taken into consideration when regularizing the social graph ($w_{ij} = C_i$). Finally, with $SN$ we denote the performance of the tree construction algorithm over the social graph exclusively, i.e., no regularization with transport network information is carried out previously.

In Figure 1, we show the overall weight of the spanning tree constructed in each of the four cases described above, as a function of the size of the social graph. First, it can be seen that the tree weights for all four cases follow a linear trend, which is expected since the size of a spanning tree increases linearly with the network size. As also expected, $SN+NET$ exhibits the most rapid trend of increase, while $SN$ exhibits the slowest, as evident from Figure 1. In particular, by taking into account the complete transport network information the tree construction algorithm is able to select the most efficient network links and graph nodes such that the overall flow of information (tree weight) is maximized. On the other hand, such
enhanced operation is clearly missing when only the social graph is taken into consideration.

Furthermore, the cases of LinkGain and UplinkCapac lie somewhere between the two extremes described above, as seen from Figure 1. Specifically, they under perform relative to SN+NET, while still providing substantial gains over SN, which justifies taking into account even partial network information when computing content delivery trees in social graphs. It should be mentioned that the relative performances of LinkGain and UplinkCapac shown in Figure 1 are due to the specific choices of uplink capacity and network cost employed in our experiments. We observed that for a different configuration of these parameters the relative performances of these two algorithms can change. Still, in all our experiments we observed that SN+NET consistently and substantially outperforms the other three cases under consideration, while the converse is always true for SN.

Finally, the overall tree weights for LinkGain, UplinkCapac, and SN shown in Figure 1 correspond respectively to 65%, 52%, and 31% of that for SN+NET, on the average.

In the remainder of the section, we explore the performance of our system against two other techniques for content delivery. The first scheme is denoted Flood as it corresponds to what is typically done today for content sharing in social networking applications. That is the content is simply flooded to all neighbours of a node that have not yet received it, starting from the content creator as the root of the flooding tree. The other reference scheme is denoted MST and it is more intelligent in its operation. This technique builds a maximum spanning tree over which the content is delivered across the social graph, again starting from the content creator as the source of the tree. When constructing its tree, MST only considers the network cost associated with each link. Via MST we can examine the additional benefits that our approach achieves by integrating in addition node weights into the tree construction procedure, as explained in Section 2. Therefore, in the following we denote the performance of our system as NW-MST.

First, in Figure 2 we briefly go over representative tree constructions on a small graph comprising 12 nodes that illustrate the design principles of each of the three approaches under comparison. It can be seen that Flood constructs the shortest delivery tree, as expected due to the way it operates. On the other hand, MST constructs the longest tree which is also expected as it attempts to maximize the overall weight of the tree, with no further consideration as to its form. The proposed technique strikes a balance between the two, as seen from Figure 2a. In particular, NW-MST tends to include the members of the social graph with higher uplink capacities as inner nodes of the constructed tree and in addition to assign more children to them. Indirectly, this makes the resulting tree shorter, relative to MST, and more importantly allows for enhanced data delivery performance, as the nodes with higher serving capacity are always employed as parent nodes. This last point is well illustrated in Figure 2 where we can see that in the case of NW-MST the Ethernet nodes (denoted in red) are all included as inner nodes of the tree, serving other Ethernet or cable/DSL peers. However, this is not the case for Flood and MST, as seen from Figures 2b and 2c, respectively.

To quantify the above comparison, we show in Figure 3 the average inner weight of the tree, per node, and the average link weight of the tree, again per link, achieved in the case of each of the three approaches, as a function of the size of the social graph. When computing these quantities, a node in a tree is characterized with its uplink capacity as a weight. Similarly, each link comprising the tree is characterized with its network cost as a weight. As seen from Figure 3a, Flood and MST achieve much smaller average node weights relative to NW-MST that in addition are roughly the same in value for the two approaches and remain constant as the network size is varied. This is expected due to the fact that both Flood and MST do not take into consideration any information associated with the nodes of the graph in their operation, as explained earlier. We can see from Figure 3a that NW-MST achieves around 26% improvement in average inner node weight by integrating such information into its design.

On the other hand, MST outperforms the other two techniques in regard to average link weight, as shown in Figure 3b. This is also expected, since due to its design NW-MST prioritizes the inclusion of links associated with higher capacity nodes into the constructed delivery tree. Such decisions can penalize the overall cost efficiency of the tree, as they may not always lead to the optimal link choice. Still, by the virtue of the fact that network costs are also considered in the operation of NW-MST, its drop in link weight performance relative to MST is relatively small (5% on the average), as seen from Figure 3b. Finally, Flood shows the worst performance also here, because it is both network cost and node capacity agnostic when constructing its multicast tree. In particular, both NW-MST and MST provide a 100% reduction in network cost relative to Flood, as Figure 3b shows.

Next, we examine the timeliness of the data delivery process associated with the three systems under comparison. In particular, we study the distribution of node delay when data is being served to the nodes of the social graph under NW-MST, Flood, and MST. In these experiments, the latter two systems employ a uniform allocation of the uplink capacity of a node when delivering data to its children in the tree. In Figure 4, we show the resulting average node latency in receiving data from the source and its standard deviation. As expected, by maximizing the inner node weight of its tree and subsequently by applying the rate allocation procedure from Section 4 to serve data over it, NW-MST substantially outperforms the two refer-
We have designed a framework for optimized sharing of user-generated content in online social networks. Our approach comprises three synergistic strategies that are sequentially applied towards the desired goal. First, we regularize the social graph by weighting its edges with data network information comprising the nodes’ uplink capacities and the edges’ data transport costs. Next, we compute a maximum weight spanning tree over the regularized graph featuring the source of the content as the root of the tree. Such a content delivery tree minimizes the network cost of sharing data between its nodes. The last technique that we apply computes the optimal distribution of the nodes’ uplink capacities across its children in the delivery tree so that additionally the data delivery delay across the tree is minimized. By the application of our framework, significant performance gains in terms of network cost and data delivery delay are registered over a commonly employed technique of flooding the content across the social graph.

6. CONCLUSIONS

We have designed a framework for optimized sharing of user-generated content in online social networks. Our approach comprises three synergistic strategies that are sequentially applied towards the desired goal. First, we regularize the social graph by weighting its edges with data network information comprising the nodes’ uplink capacities and the edges’ data transport costs. Next, we compute a maximum weight spanning tree over the regularized graph featuring the source of the content as the root of the tree. Such a content delivery tree minimizes the network cost of sharing data between its nodes. The last technique that we apply computes the optimal distribution of the nodes’ uplink capacities across its children in the delivery tree so that additionally the data delivery delay across the tree is minimized. By the application of our framework, significant performance gains in terms of network cost and data delivery delay are registered over a commonly employed technique of flooding the content across the social graph.

7. REFERENCES