Term Project Report

Intelli Cache

UNDER THE GUIDANCE OF:

Mr. Sandip Chakraborty

(Assistant Professor)

Mr. Satadal Sengupta

(JRF and MS Student)

SUBMITTED BY:

Shravan Kumar Singh (16CS60R79)

Gaurav Singh (16CS60R64)

Rahul Rank (16CS60R44)

Omkar Sharan (16CS60R28)

Department of Computer Science & Engineering
I.I.T Kharagpur

1. Problem Definiton

Content centric network employs the concept of *in-network management* of content storage and distribution, where intermediate routers or base stations apply content storage and distribution policies based on the underlying content usage patterns. CCN uses content popularity as basic criteria for deciding content storage, and distribution policies. There is a direct correlation between content popularity and Social dynamics among users of that content.

IntelliCache is a cache implementation, which runs on a proxy server. Based upon the user access pattern it develops a directed graph G which represents the social dynamics between users. Now *IntelliCache* uses this graph G to infer the content popularity and then uses this content popularity measure to take caching decision.

2. Solution Approach

1. Packet Trace Collection: This is done at the proxy server through which all the traffic is passing.
2. Events : Only two types of events are considered namely sharing and viewing.

   a. Viewing
   Identification: video streaming traffic exhibits a distinct bursty pattern with troughs in between bursts.

   Action:
   ADD <content_id, user_id, event, timestamp> to the event timeline.

   where,
   content_id: Unique string that identifies the video that is being streamed. This can be obtained from domain name system (DNS) query which is sent by the user to get the URL of the video.
   user_id: Identification of the user watching the video.
   event: *VIEW*
   timestamp: Time at which the event took place.

   b. Sharing
   Identification: Whenever share button is clicked Facebook makes a call to graph.facebook.com

   Action:
   ADD <content_id, user_id, event, timestamp> to the event timeline.

   where,
   user_id: Identification of the user watching the video.
   content_id: Unique string that identifies the video that is being shared. Assumption is that the user will share the video within 20 seconds after the streaming has been ended. So to get the content_id we backtrack in the event timeline and look for the last video that is viewed by the user_id.
   event : *SHARE*
   timestamp : Time at which the event took place.

3. Event Tree Generation

   a. Event Tree is created for every content_id.
   b. Every useri_id that *SHARE* or *VIEW* content_id becomes a node of the event tree for that content_id.
   c. If *A* is the parent of node *B, it* means that *B* views the video shared by the node *A.*
   d. If A has only viewed but not shared then it will become leaf.
   e. Multiple view of the same video will not affect our tree.

   The above process will generate a forest that will be input for the next stage.

4. Social Graph Estimator

   Output of this step is a directed weighted graph $G_{Inferred}$ which estimates the social dynamics between the users. Nodes in graph $G_{Inferred}$ are the nodes which appeared in at least once in any of the event trees. Following Algorithm is used to generate $G_{Inferred}$.

Input: List of all *Event Trees*, Threshold edge confidence $Conf_{threshold}$
Output: Estimated Social Graph $G_{inferred}$
$EventForest$:= List of all *Event Trees*;
$G_{intermediate}$:= $NULL$;
while $EventForest$ *is not empty* **do**
  $CurrentEventTree$:= The first *Event Tree* in $EventForest$;
  foreach $DirectedEdge \in CurrentEventTree$ **do**
    if $DirectedEdge \in G_{intermediate}$ **then**
      Increment weight of corresponding directed edge in $G_{intermediate}$ by 1;
    end
    Add corresponding directed edge to $G_{intermediate}$ with weight 1;
  end
  Delete $CurrentEventTree$ from $EventForest$;
end
$G_{inferred}$:= $NULL$;
foreach $DirectedEdge \in G_{intermediate}$ **do**
  $Sum_W = 0$;
  foreach $OutgoingEdge$ *from node on which* $DirectedEdge$ *is incident* **do**
    $Sum_W := Sum_W + W_{OutgoingEdge}$
  end
  $Conf_{DirectedEdge} := W_{DirectedEdge}/Sum_W$;
  if $Conf_{DirectedEdge} > Conf_{threshold}$ **then**
    Add $DirectedEdge$ to $G_{inferred}$;
  end
end

Input to the following algorithm is the list of all the event trees generated during the *Event Tree Generation process*. The threshold value $Conf_{threshold}$ determines whether an edge should remain in the graph $G_{Inferred}$ or should be dropped as a coincidental edge i.e. if weight of an edge in graph $G_{Inferred}$ is less than $Conf_{threshold}$ then that edge will be dropped from the graph $G_{Inferred}$.

5. Centrality Measure

We use an index [4] which considers both the number of edges and strength of an edge for finding degree of centrality of a node. This measure gives us a way to find the node which is most influential in the network. Here is the index:

$$C_D^{w\alpha}(i) = k_i \times \left(\frac{s_i}{k_i}\right)^{\alpha} = k_i^{(1-\alpha)} \times s_i^{\alpha}$$

Where,

1. $k_i = \sum_j^N x_{ij}$, i is the focal node**,** j represents all other nodes, N is the total number of nodes, and x is the adjacency matrix, in which the cell x_{ij} is defined as 1 if node i is connected to node j, and 0 otherwise.
2. $s_i = \sum_j^N w_{ij}$, where w is the weighted adjacency matrix, in which w_{ij} is greater than 0 if the node i is connected to node j, and the value represents the weight of the tie.
3. α is a positive tuning parameter. If we want to favour high degree we would choose α between 0 and 1 and for low degree we would choose α greater than 1.

Effect of α on the value of this measure for the nodes of the network is shown in Figure 1. As shown by table, when $\alpha = 1$ the measure's value equal the node's strength. When $\alpha < 1$ and the total node strength is fixed, the number of contacts over which the strength is distributed increases the value of the measure. For example, when $\alpha = 0.5$, node B attains a higher score than node A, despite having the same node strength. Conversely, when $\alpha > 1$ and the total node strength is fixed, the number of contacts of which the strength is distributed decrease the value of the measure in favour of a greater concentration of node strength on only a few nodes.

Node	C_D	$C_D^{w\alpha}$ when α =			
		0	0.5	1	1.5
A	2	2	4	8	16
B	4	4	5.7	8	11.3
C	2	2	3.5	6	10.4
D	1	1	1	1	1
E	2	2	4	8	16
F	1	1	2.6	7	18.5

Table 1

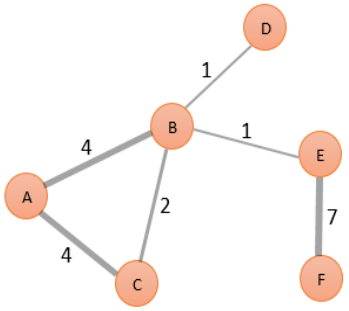
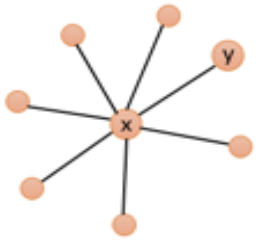


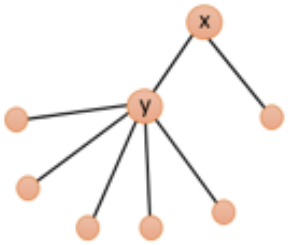
Figure 1

3. Findings

1. Proposed solution is not considering whose shared video is being viewed. Let's assume person x is an influential person with very large social group and shares a video. His friend/follower "y" watches the video and shares it immediately. Now if other people watch this video, person x will not get the weights in the inferred Graph because an edge will be drawn from y to other people, and not from x.



Original graph



Estimated graph

2. If user watches a video and does not share it right away. Then goes down in the Facebook page and watches other video, and then he decides to share the previous video. In this case how we would know what video we are sharing because the link that we have got right now is of last played video. In this case we would cache the video which we played at last but not of the video that we played in starting.

4. Work Distribution

1. Findings:Shravan Singh, Rahul Rank
2. Report Preparation: Gaurav Singh, Omkar Sharan
3. Analysis of paper:Shravan Singh, Gaurav Singh, Rahul Rank
4. Coding: Gaurav Singh, Rahul Rank, Omkar Sharan

5. References

1. Centrality. n.d. In *Wikipedia.* Retrieved March 13, 2017, from https://en.wikipedia.org/wiki/Centrality
2. pyshark 0.3.7.2 .n.d. In *Python.* Retrieved March 13, 2017 https://pypi.python.org/pypi/pyshark
3. How to Use Wireshark to Capture, Filter and Inspect Packets. October 14, 2014. In *howtogeek.* Retrieved March 13, 2017 https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/
4. Opsahl, Tore, Filip Agneessens, and John Skvoretz. "Node centrality in weighted networks: Generalizing degree and shortest paths." *Social networks* 32.3 (2010): 245-251.