International Journal of Electronics Engineering (ISSN: 0973-7383)
Volume 11 • Issue 1 pp. 571-575 Jan 2019-June 2019 www.csjournals.com

Traffic Load Distribution in Binary Tree network

Ruchi Sodhi[a], Ghanendra Kumar[b] and Chakresh Kumar*[a]
[a]University School of Information, Communication & Technology, Guru Gobind Singh IP University, New Delhi
[b]Department of Electronics and Communication Engineering, National Institute of Technology, Delhi-110040, India
*mail-id: chakreshk@gmail.com

Abstract—Congestion in any network happens when a given node or connection/edge is carrying more data than it is capable of. This results in decreased quality, as the reliability of data transmission could be compromised. The congestion can happen in any network topology, and binary tree networks are no exception. Congestion is bad for a reliable network as it could result in delayed delivery and lose of data packets. We are going to look at ways of how congestion can be reduced in a binary tree network, so that no node in under a load which is more than it can handle.

Keywords-Topology, binary, data rate, point to point.

I. INTRODUCTION

The binary tree as a network topology provides us with many benefits, when implementing a peer to peer system. In case of a balanced tree network, the maximum number of hops is no more than 2* Log2 (n), where n is the total number of nodes. This is an improvement over topologies like ring, where the number of hops could reach n/2. Despite this benefit, there is a major disadvantage which binary-tree network brings in. There might be a case when the data traffic flowing through a node is very high, causing it to lose some packets or even crash. Consider a simple example where 10 nodes in the left sub-tree of the root are requesting data from 10 different nodes in the right subtree of the root. All this traffic will have to pass through the root. This could be true for any node, not just root[1-2].

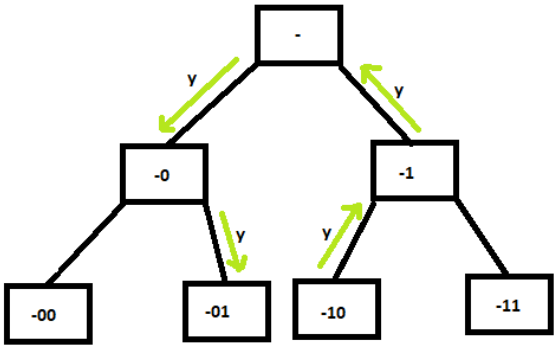


Fig. 1 Single data streaming at a time

The above diagram describes a scenario where only one video streaming process is taking place at a given time. **NODE-01** asks for a Video-Y on the network. For simplicity, let us assume that only one node **NODE-10** hosts that video[4-5]. When the video starts streaming, **NODE-10** will start delivering the video packets. The rate at which the data will be transmitted will depend on the quality of the video, or in simpler words total size (in bits)/ length(in seconds) of the video will be transmitted from one node to another per second. So for our scenario, y bits will be transmitted from **NODE-10** to **NODE-1**, from **NODE-1** to **NODE-**, from **NODE–** to **NODE-0** and finally from **NODE-0** to **NODE-01**.

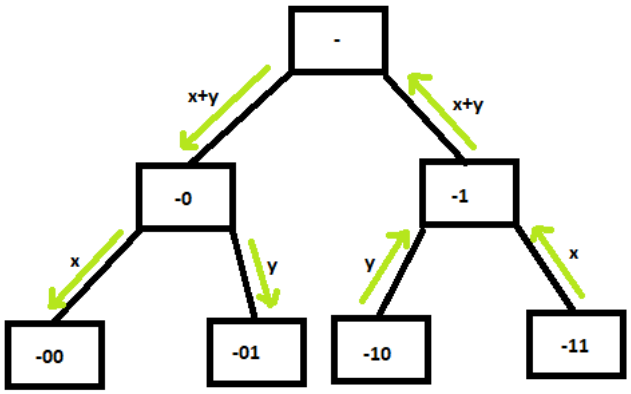


Fig. 2 Simultaneous data sharing

Now, let us consider a realistic scenario where more than one node/user wishes to stream a video at a given time. In the above diagram, node **NODE-00** requested a Video-X that is hosted at node **NODE-11** and node **NODE-01** requested another Video-Y that is hosted at **NODE -10**. For Video-X, the data transmission requires is x bits/sec. and for video-Y, it is y bits/sec. From **NODE-1** to **NODE–** and from **NODE–** to **NODE-0**, the data transmission rate will be the sum of the transmission rate of the 2 videos, i.e. **(x+y) bits/sec**. This is not a big issue for few nodes, but as the network grows bigger and bigger, it will cause the data transmission rates to exceed what is supported by any network and could lead to packet loss or node **NODE–** to crash(depending on the implementation). This case is applicable to any node in the network as any node can act as a root to a subtree. This introduces reliability issues in our network which is undesirable for any peer-to-peer network.

To overcome this issue, we could use various mechanisms. We can ask the streaming node to slow down the streaming of data. This could be useful form some scenarios like file transfer, but cannot be used for video streaming or video call. For real-time delivery of data, we could use mechanisms like delivering less quality video. We can also introduce in some transient edges directly between the nodes to reduce the choking of the network. We are going to discuss the following approaches as solution to our problem.

1) *Throttling*
2) *Adaptive resolution.*

1) Throttling:

Data transmission throttling is technique where we intentional slow down or speed up delivery of data packets by asking the data generation computer to produce data slowly. It technique is reactive in nature, which means that the slowing or speeding happens when the congestion starts to happen and helps to regulate data traffic and avoid the implications of congestion, i.e. data loss or node crash. Data transmission throttling can be introduced at any node/link on the data network. This technique can be used in any type of topology and each network can have its own way of introducing it. The network could be smart enough to detect congestion as request the data generating node to slow down or a system admin could throttling down the node by limiting the upload speed of the node to help avoid node crashes. To generalize, any network can use data transmission intentional slowing down/speeding up to help improve the reliability of the data transmission capability of a network.

Data transmission throttling can be used to reactively limit a node's streaming up or streaming down speed. This technique can be is useful where a delay in delivery will not affect the business. For example, it is acceptable in case of file download, where the download speed goes down momentarily. But for the case of real-time activities like video streaming or video call, this could introduce a lag or delay in the delivery. So, this kind of technique is not applicable for such cases. Many protocols like bit-torrent and distributed file sharing systems use this technique to limit their transfer speed to avoid network issues. Data throttling technique is also used in network apps, to distribute the load over a broad network and to bring down the network data congestion or distribute over a cluster of nodes to avoid overloading on any node thereby eliminating their chances crashing the network.

Throttling in binary tree:

As discussed earlier, sue to streaming of multiple data simultaneously, the root node (or any other) could crash or there could be loss of data packets. To avoid such problems, we could introduce throttling in our binary-tree network. The throttling could be done by sending a message to the data producing node to produce data at a lower rate. This notification will be sent from the node, which is currently under load, to the nodes which are producing data at an alarming rate. The definition of 'load' may vary. But for simplicity, we can use a threshold data transmission rate, beyond which we will say that the node is under load. When this threshold crosses for a given node, that node will send a slowdown message to the data producing node.

We also have to consider the fact that one the load goes down the threshold, we have to tell the slowed down node to speed up, so that it can work to its full potential. This is again, the responsibility of the node that was previously under load.

Message Type:

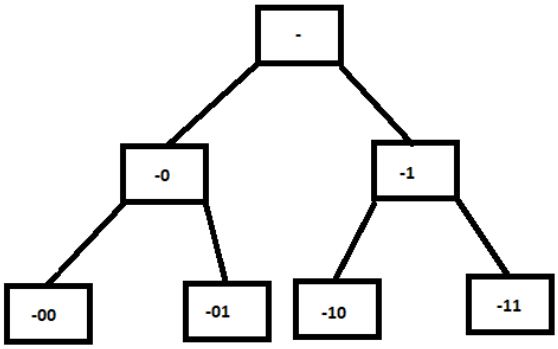
i) **SLOW_DOWN:** The message is sent by the node under load to the node that is producing data. It will be a predefined message, it will contain the logical address of data producing node along with the rate by which it expects it to slow down. This value will be decided by the difference in the current load and the threshold to bring the load under the threshold.

ii) **SPEED_UP:** The message is sent by the node that was under load previously, but now it isn't. It is sent to the node that, that was previously asked to slow down. This is also a predefined message data. It will contain the logical address of data producing node along with the rate by which it expects it to speed up. This value will also be decided by the difference in the current load and the threshold to bring the load under the threshold.

We can see how throttling will work by using an example.

Illustration:

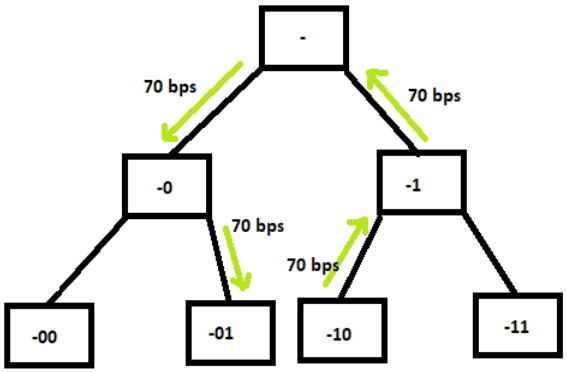
Consider the following balanced binary-tree network that exist with7 nodes connected.



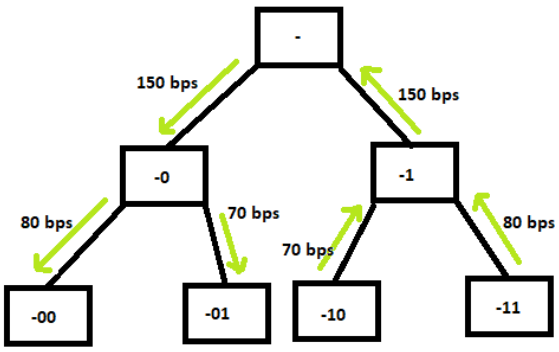
In the above network, there are 2 files **FILE-A** and **FILE-B**. **FILE-A** is hosted on **-11** and **FILE-B** is hosted on **-10.** The transmission threshold is set to be **100 bits/sec**.

a) **NODE-01** wants to download **FILE-B**, that is hosted at **NODE–10**. **NODE-01** sends a **SEARCH_FILE_REQUEST** message to the entire network(broadcast), with FileName as **FILE-B**. Only **NODE-10** send a response to **NODE-01** since it is the only one with that file. After receiving the response from **NODE-10**, **NODE-01** makes a download request toNode **-10** for **FILE-B**. Now, the file data packets are delivered from **NODE-10** toNode **NODE-01** such that the transmission rate

is 70 bits/sec. The packets are delivered through nodes **NODE-10**, **NODE-1**, **NODE-**, **NODE-0** and finally **NODE-01**.



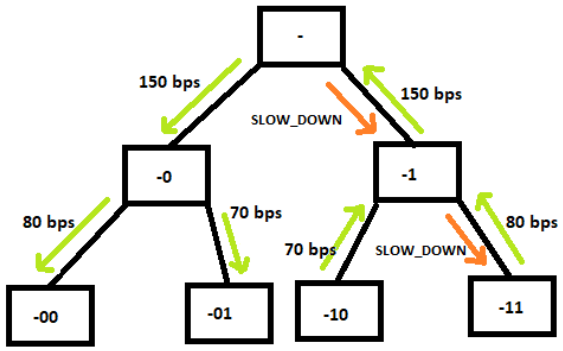
b) **NODE-00** wants to download **FILE-A**, that is hosted at **NODE–11**. **NODE-00** sends a **SEARCH_FILE_REQUEST** message to the entire network(broadcast), with FileName as **FILE-A**. Only **NODE-11** send a response to **NODE-00** since it is the only one with that file. After receiving the response from **NODE -11**, **NODE-00** makes a download request to **NODE-11** for **FILE-B**. Now, the file data packets are delivered from **NODE-11** to **NODE-00** such that the transmission rate is 80 bits/sec. The packets are delivered through nodes **NODE-11**, **NODE-1**, **NODE-**, **NODE-0** and finally **NODE-01**. But since FILE-B is already being transferred, the transfer rate of data will be 150 bits/sec on nodes **NODE-1**, **NODE-**, **NODE-0**.



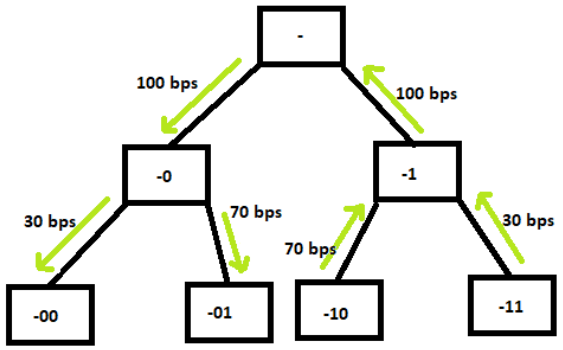
c) Since 150 bits/sec exceeds the given threshold(100bits/sec), a throttling request will be made by the **NODE-1** to **NODE-11**, since it is producing the maximum amount of data. The message sent will be

{DestinationNode:-11, MessageType: SLOW_DOWN, reduceBy: 50bps, transferId:0231X1}

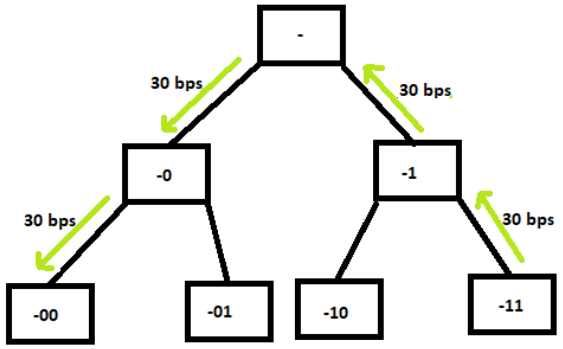
The above message means that **NODE-11** is requested to slow down the data transfer by 50 bits/sec.



The **NODE-11**, on receiving this message reduces the transfer rate from 80 bps to 30 bps. **NODE-11** decides which data transfer to slow down, based on the transfer Id. The transfer id will be unique for a given download and will also be used when requesting to speed up.
After slowing down, the load through the nodes **NODE-1**, **NODE and NODE-0** is 100 bps.

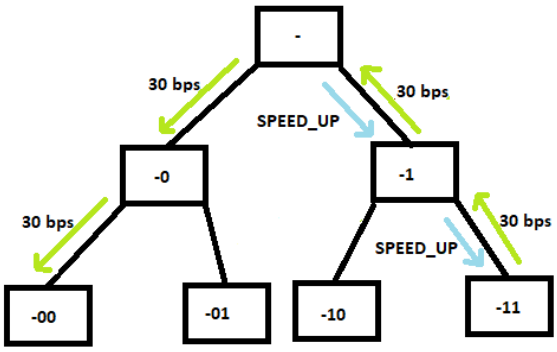


d) Once the trasnmission of **FILE-B** is complete, the transfer rate through nodes **NODE-1**, **NODE-**, **NODE-0** falls to 30 bps, which is much lower that the threshhold(100bps). Now the network is not using its full bandwidth and causing unnecessary delay.

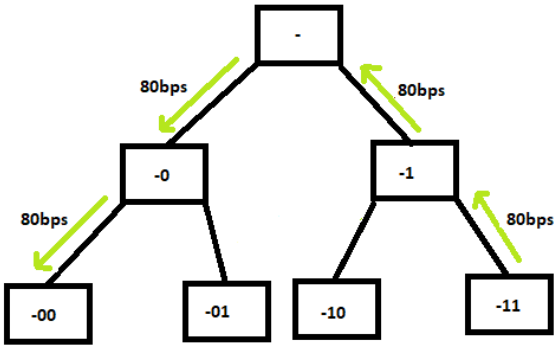


e) Now, the **NODE-** will send a SPEED_UP message to the producer **NODE-11** to speed up the ransfer of data as **NODE-** knows that it is not delivering data to its full potential. The following message will be sent from **NODE-** to **NODE-11**.

{DestinationNode:-11, MessageType: SPEED_UP, increaseBy: 50bps, transferId:0231X1}



f) This message will result in **NODE-11** to increase its transfer speed back to 80bps. As a result the network will be back to working at its full potential.



The transfer will happen at this speed till the transfer is complete or till no other throttling is introduced.

2) Adaptive resolution:

Throttling is very efficient when it comes to transferring data with high reliability, but it could also introduce some delay. So, for a system requiring real-time data transfer, throttling alone could not be the best approach. There as cases like on demand video or video/audio call where our main focus is to achieve a real-time user experience. In such cases we can compromise the quality of the data being transferred to overcome the reliability issues and delays. For such cases, we can choose to send data with low quality, till the time the nodes on the path of data transfer are under high load. For Video transfer, we can store video frames with multiple resolutions. Whenever a node on the path of transmission is under load, that node will request the producer node to deliver video frames of lower resolution. Once the load reduces the given threshold, the node that was previously under load could request the data producing node to transfer high resolution video frames again.
The following message types will be used for increasing or decreasing resolution.

Message Type:

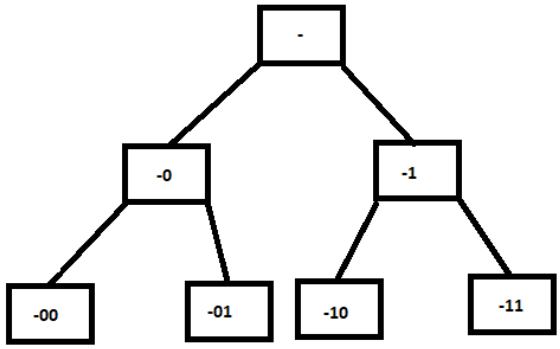
iii) **INCREASE_RESOLUTION:** The message is sent by the node under load to the node that is producing data. It will be a predefined message and will contain the logical address of data producing node.

iv) **DECREASE_RESOLUTION:** The message is sent by the node that was under load previously, but now it isn't. It is sent to the node that, that was previously asked to produce low resolution data. This is also a predefined message and contains the logical address of data producing node.

We can see how adaptive resolution will work by using an example.

Illustration:

Consider the following balanced binary-tree network that exist with7 nodes connected.

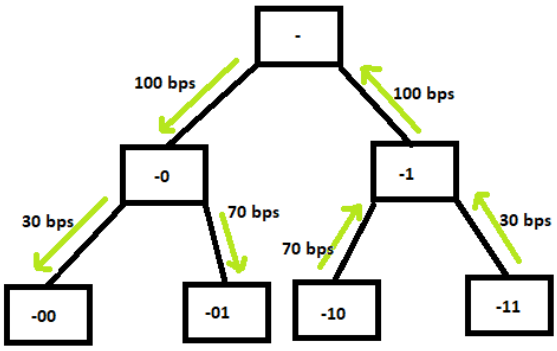


In the above network, there are 2 files **VIDEO-A** and **VIDEO-B**.
VIDEO-A is hosted on **NODE-11** and **VIDEO-B** is hosted on **NODE-10.** The transmission threshold is set to be **100 bits/sec**.

The idea is similar to what is described in the throttling technique. The difference is that rather the transmission rate exceeds the threshold; the node under load will ask the node sending the maximum resolution data to send lower resolution data. This is done by sending DECREASE_RESOLUTION message to that node.

{DestinationNode:-11, MessageType: DECREASE_RESOLUTION, transferId:0231X1}

Use of lower resolution data will automatically reduce the load on the link/node(low resolution takes less space).



Once the load reduces from the threshold, the node that was under load previously will ask the node that is sending low resolution data to increase ther resolution. For that an increase message is sent to the data producing node.

{DestinationNode:-11, MessageType: INCREASE_RESOLUTION, transferId:0231X1}

On receiving this message, the node will transfer data to its full potential.

REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to Algortihms(2010)

[2] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Germany (Print ISBN 978-3-540-42800-8)

[3] Detti A., Pomposini M., Blefari-Melazzi N., Salsano S., and Bragagnini A., (2012) “Offloading cellular networks with information-centric networking: The case of video streaming,” in 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), (pp. 1–3).

[4] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Newsletter ACM SIGCOMM Computer Communication Review – Proceedings of the 2001 SIGCOMM conference (Volume 31 Issue 4)

[5] Harami C., Gazdar A., Jamelli I., and Belgith A., (2015) "Study of VOD streaming on BitTorrent," IEEE International Symposium on Networks, Computers and Communications (ISNCC).