DESIGN OF CROWD-SCALE MULTI-PARTY TELEPRESENCE SYSTEM WITH DISTRIBUTED MULTIPOINT CONTROL UNIT BASED ON PEER TO PEER NETWORK

A dissertation submitted

to Kent State University in partial

fulfillment of the requirements for the

degree of Doctor of Philosophy

by

Md Amjad Hossain

December 2020

Dissertation written by

Md Amjad Hossain

B.S., Khulna University of Engineering and Technology, Bangladesh 2008

Ph.D., Kent State University, 2020

Approved by

Dr. Javed I. Khan ____________________, Chair, Doctoral Dissertation Committee

Dr. Cheng Chang Lu _________________, Members, Doctoral Dissertation Committee

Dr. Gokarna P. Sharma________________

Dr. Murali Shanker___________________.

Dr. Jun Li___________________________.

Accepted by

Dr. Javed I. Khan_____________________, Chair, Department of Computer Science

Mandy Munro-Stasiuk, Ph.D. __________ , Interim Dean, College of Arts and Sciences

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DEDICATION

This work is dedicated to my encouraging parents, brothers, sister, supportive and loving wife Preoyati khan, and our sweet daughter Alisha Hossain.

ACKNOWLEDGEMENTS

Thank you, GOD, for helping me to complete this dissertation. I would like to express my gratitude and appreciation to my advisor, *Dr. Javed I Khan*, for his expert guidance and mentorship, his brilliant and thought-provoking ideas, his encouragement and support at all levels. I would also like to thank the defense committee members for their constructive comments and much-appreciated advice. I would like to thank every one of the Computer Science department who supported me in this long journey, especially *Marcy Curtiss* and *Janet Katila*.

Finally, I would like to extend my sincere gratitude to my parents, my wife, my daughter, and the rest of my family for their patience, continuous support, and encouragement.

Md Amjad Hossain

August 2020, Kent, Ohio

CHAPTER 1

Introduction

1.1 Background

This dissertation presents protocols for peer-to-peer scalable telepresence systems. The telepresence system refers to a set of technologies that allow the users to feel as if they are physically present at a place other than their true location[1-4]. These technologies facilitate sensing data for different human sensors, including vision, sound, smell, taste, touch, etc., and transport them to the interested users. The technological development in the last two decades has enabled the implementation of the Multi-Party Telepresence System(MTS), where everyone can receive all other participants' audio-visual streams and other sensory data. The most popular class of MTS is Multi-Party Video Conferencing(MVC), where the users can exchange audio-video signals among them and see each other [5-8]. The MVC systems have experienced significant changes over the last couple of decades. Starting from the studio-based solution, which requires dedicated ISDN lines, expensive equipment, etc., now the MVC applications are in personal computers, smartphones, etc. The MVCs have evolved with the maturity of packet-switched or IP network, encoder/decoder(CODEC)[9, 10], signaling protocols[11], etc. Currently, most of the commercial MVC applications are supported by cloud servers instead of dedicated devices.

Traditionally, video conferencing has been within a limited number of users. With the development of the MVC systems, their demand has also significantly increased. Large-scale MTS or the MVC is required for arranging many mega-events virtually. The example of mega-events includes parliamentary meetings, tech-conferences, summit meetings, industrial meetings, etc. The number of participants in these events can be several hundred to thousands. As the participants attend these events physically, it makes those events expensive to organize and participate. For international events, participants need to go through complicated, time-consuming processes, including visa approval. If the event is a multiparty political meeting, some participants might feel unsafe to attend the sessions physically. These events can be cost-effective, safe, and hassle-free if virtual participation can be enabled using a Crowd Scale MTS. Such a system is also useful in global pandemic situations caused by viruses such as Ebola, COVID-19, etc. While people maintain social distancing to reduce the spread of germs, they can still attend different important mega-events, remote classes, virtual gatherings, etc. Research shows that the lockdown caused by COVID-19 has lowered carbon emission[12, 13]. So, the remote participation (staying home) through MTSs can reduce the global indicator of carbon footprint(CF). In this dissertation, an architecture of Crowd-scale MTS is discussed.

1.2 Typical Architecture of MTS

As mentioned earlier, most of the MTSs are in video conferencing classes, and they run on one of the following two architectures:

a) *Mesh Network:* In this architecture, the participant nodes form a peer to peer(P2P) overlay network, which can be full or partial mesh[14, 15]. A tree-based transmission model is used to transfer the video stream of a participant to others. So, each participant must create an application layer multicast(ALM) tree [5, 16, 17]. Therefore, each participant needs high bandwidth and computing power to receive and process all the streams from other participants. Such an architecture may or may not include a central server for session management[18].

b) *Star Network*

   i. *Centered at Multipoint Control Unit(MCU):* In this architecture, the MCU receives streams from all participants, combines them into one, and sends them back to all participants [19]. Traditionally, the MCU has been expensive dedicated hardware with a limited number of ports. In recent years, the software MCUs are available, and they can be placed in one or more dedicated servers (can be from the cloud) to form the central MCU[20-22]. The MCU can allow participants with relatively low bandwidth to attend the conference.

   ii. *Centered at Selective Forwarding Unit(SFU):* The SFU works based on the concept of Simulcast[23], where each participant generates multiple streams of different bitrates and sends them to SFU. The SFU forwards some selected streams to others based on their available bandwidth. The actual stream processing is done at the participant nodes. So, each participant must have the computing power for processing and high bandwidth for transmitting and receiving multiple streams to and from the SFU[24-26].

Most of the MTS services are still based on central MCU. The service providers may place one or more MCU devices statically at the conference sites or use software MCUs on the cloud. So, these systems involve one or more service providers who manage central units and have full control over the privacy and security of users. Moreover, these MCUs are expensive to organize large scale telepresence sessions.

1.3 Crowd-scale Multiparty Telepresence System(CMTS)

The traditional MTS (usually video conferencing) can accommodate a limited number of users. So, we redefine such a system as Crowd-scale MTS or CMTS. The name suggests that the system should accommodate a crowd of participants in a telepresence session. In CMTS, the participants should feel the real crowded environment, such as sitting in a large parliamentary meeting, stadium, classroom, conference room, etc. For that, the system must collect multi-channel streams of all participants and create the desired virtual layout or environment from the visual components (video or images) of the streams, mix other channels and deliver the combined stream to all the interested participants. The 5G network will make it possible to carry many streams of CMTS within the allowable communication delays. But such a system would require complex and dynamic synchronization of many multi-channel streams, including heavy-duty audio, video, or holographic encoding, decoding, transcoding, and stream remixing in real-time. The processing of volumetric video and the holographic scene would be more complicated than the 2D video scene. Because for holography, data must be sent in multiple channels for many features (such as light, air, temperature, etc.) of the environment to be transferred.

1.4 The Challenges of CMTS

Based on the requirements of CMTS, the significant challenges to implement CMTS are Visual and Cognitive, Computational, Temporal, and Overcrowding. The problems associated with these challenges and their solutions are often interrelated.

1.4.1 Visual and Cognitive Challenge

The remote participants of large events would like to see each other in a real scene or environment such as a stadium, parliament, amphitheater, etc.

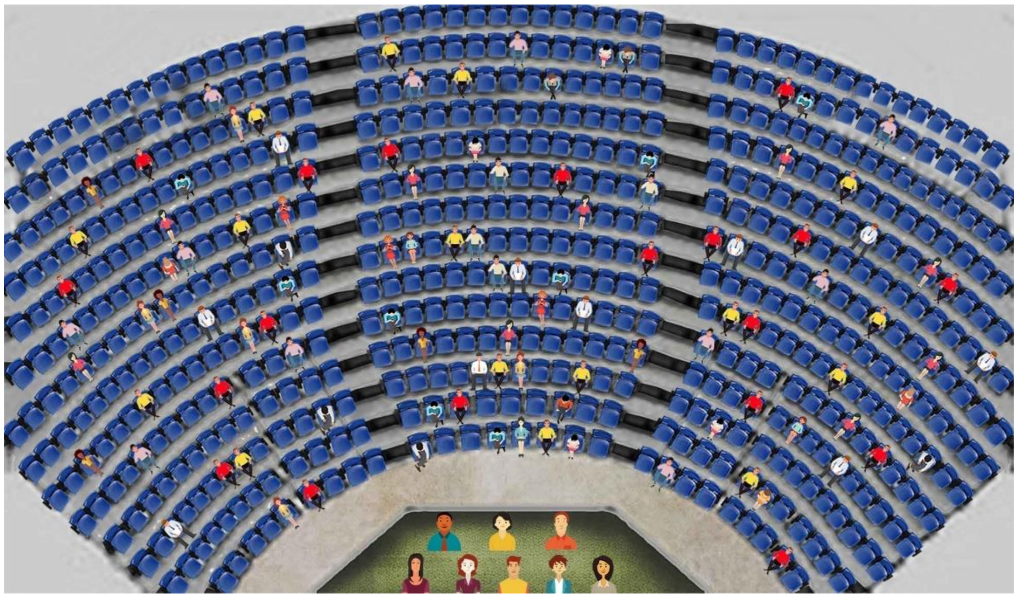


Figure 1.1 Visual Layout of CMTS

Figure 1.1 shows an example of a visual layout for CMTS, where the participants are shown in the gallery. For reducing the cognitive load, the layout must have a focused area. The participants can choose to see some close friends in the focused area, as shown in the bottom part of the layout. The participants might even want to move the focused area to different parts of the screen. However, the generation of such a dynamic layout for CMTS would require continuous matrix transformation with other complex operations. A similar layout based on holography or volumetric video would be too complex to generate.

1.4.2 Computational Challenge

MCU is the core element of any MTS. It connects all system users within a single network and provides a wide range of complex functionalities.

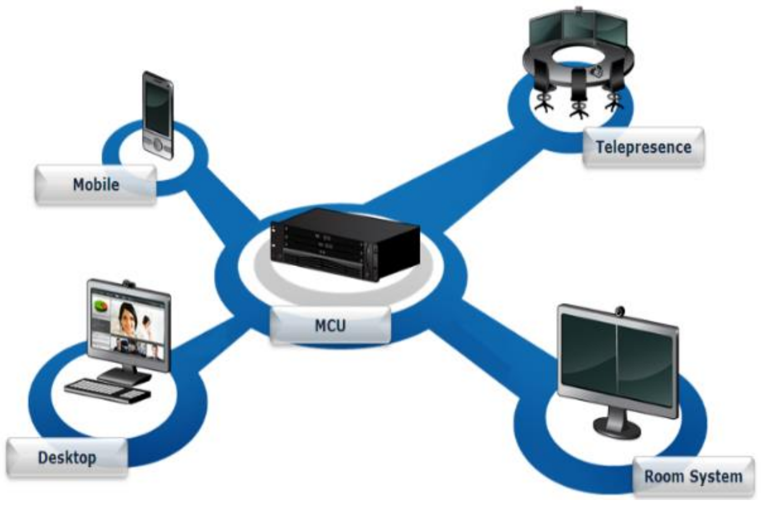


Figure 1.2 The MCU connecting multiple endpoints[27]

It can also connect another MTS to create a more extensive system, Figure **1**.**2**. The MCU consists of a Multipoint Controller(MC) and zero or more Multipoint Processors (MPs)[28]. MC works in the control plane and generates control signals for the media plane. The MPs work in the media plane, where they perform various tasks such as forwarding, switching, recording, audio, and video mixing (composing) and transcoding, etc. Among all the operations, the primary task performed by the MCU is mixing. It is the process of decoding all incoming streams (including sub-channels), combining them channel-wise, and re-encoding combined stream with inter-channel synchronization satisfying constraints on output stream rate, Figure 1.3, [19, 22, 29]. The MCU can connect participant devices that have different bandwidth and use different signaling protocols (SIP, H.323), different codecs for audio (G.711, G.722, G.722.1, etc.), and video (H.263, H.264, etc.). So, it provides necessary conversions, including transcoding and trans-rating[30]. For CMTS, providing all these functionalities of the MCU would be very challenging.

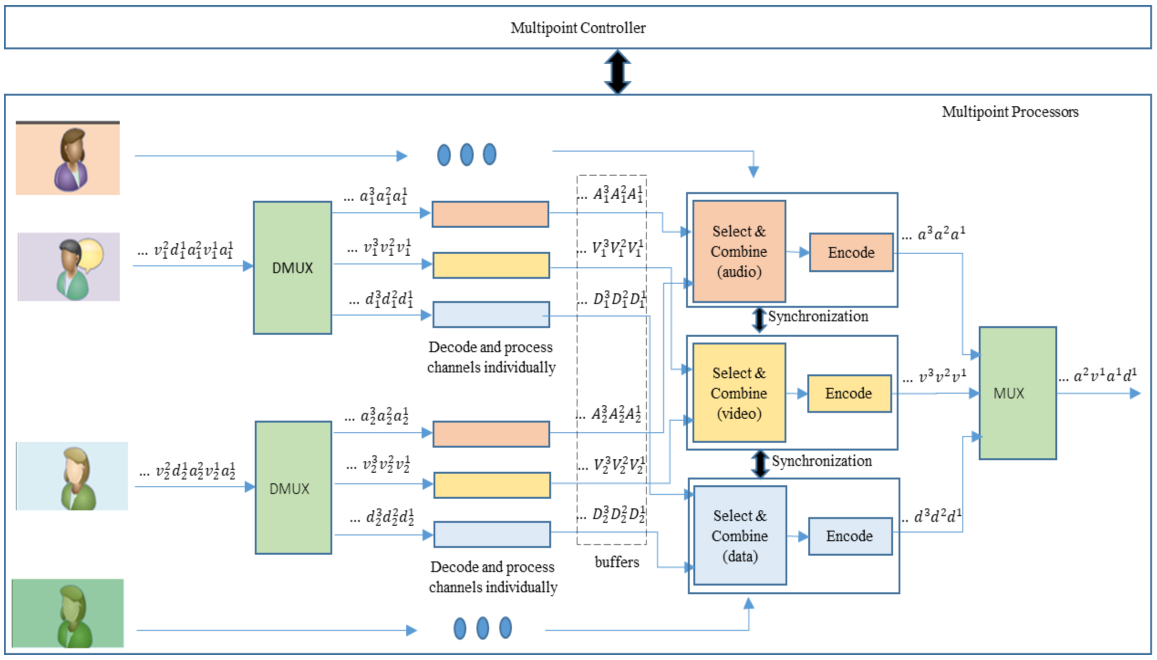
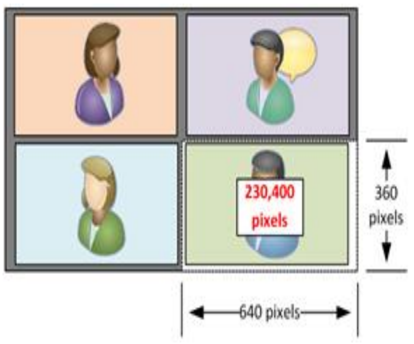
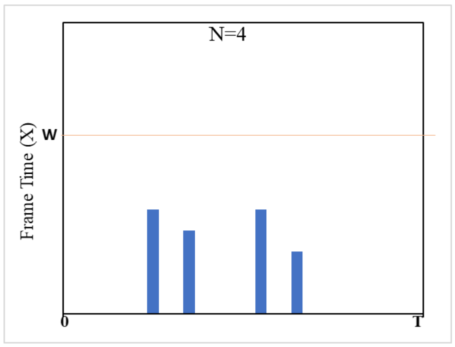

Multipoint Controller
Multipoint Processors
$\dots v_1^2 d_1^1 a_1^2 v_1^1 a_1^1$
DMUX
$\dots a_1^3 a_1^2 a_1^1$
$\dots v_1^3 v_1^2 v_1^1$
$\dots d_1^3 d_1^2 d_1^1$
Decode and process channels individually
$\dots A_1^3 A_1^2 A_1^1$
$\dots V_1^3 V_1^2 V_1^1$
$\dots D_1^3 D_1^2 D_1^1$
$\dots v_2^2 d_2^1 a_2^2 v_2^1 a_2^1$
DMUX
$\dots a_2^3 a_2^2 a_2^1$
$\dots v_2^3 v_2^2 v_2^1$
$\dots d_2^3 d_2^2 d_2^1$
Decode and process channels individually
$\dots A_2^3 A_2^2 A_2^1$
$\dots V_2^3 V_2^2 V_2^1$
$\dots D_2^3 D_2^2 D_2^1$
buffers
Select & Combine (audio)
Encode
$\dots a^3 a^2 a^1$
Synchronization
Select & Combine (video)
Encode
$\dots v^3 v^2 v^1$
Synchronization
Select & Combine (data)
Encode
$.. d^3 d^2 d^1$
MUX
$\dots a^2 v^1 a^1 d^1$

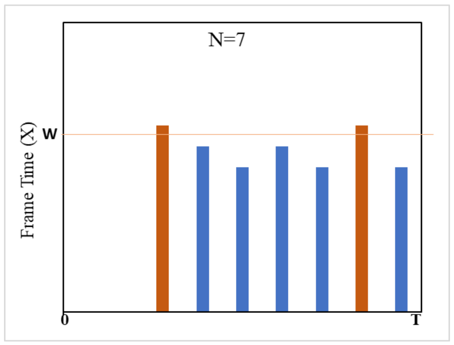
Figure 1.3 Multi-channel mixing in the MCU



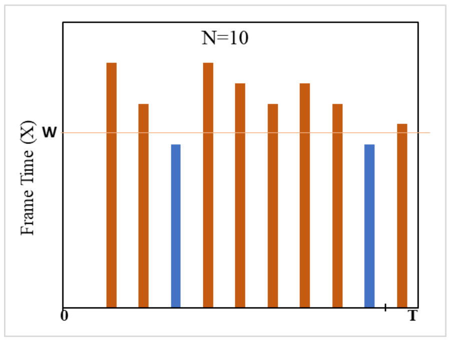
(a)



(b)



(c)



(d)

Figure 1.4 Frame composition within the given waiting time W

1.4.3 Temporal Challenge

The deployment of a hardware-implemented MCU device is complex. So, the software implemented MCU has become prevalent as it can be installed on any computer, even in the cloud servers. The MCU sits at the center of the star topology of an MTS, Figure **1**.**2**. So, the number of participants is limited by the available bandwidth and the computing power of the server running the software MCU.

The human visual system can process 10 to 12 images per second and perceive them individually, while higher rates are seen as motion [31]. So, for motion perception, the frame interval must be less than ≈100ms. Suppose in an MCU-based MTS; there are *n* participants. The expected frame interval is given, which is *T* milliseconds ($\frac{1000}{T}$ frame rate). The participants generate frames with the same interval/rate so that the MCU can generate a composite frame, as shown in Figure 1.4(a) after each T milliseconds. A participant's frame travels to the MCU and may need to wait in the buffer before being inserted into the composite frame, Figure 1.3. Suppose, for node i, x_i is the frame time, which is the time duration between the frame is generated from node *i,* and it is inserted in the composite frame by the MCU. So, $x_i = d_i + b_i + \delta$, where d_i is the end-to-end delay from node *i* to the MCU, b_i is the time spent by the frame in the buffer, and δ is the constant time taken by the MCU to mix the frame in the composite frame. For inter-stream synchronization, the MCU doesn't insert a frame in the composite frame if its frame time exceeds a tolerable threshold, say *W*. So, a frame of node *i* will be inserted in the composite stream only if its frame time $x_i \leq W$, otherwise it is dropped. The frame time will be random because of uncertain link and buffer delays. Suppose the frame times of participants are represented by a normal random variable X with mean μ and standard deviation σ. Then the probability that a participant's frame will be inserted (only if the frame time $X \leq W$) in the composite frame is,

$$P(X \leq W) = \int_0^W f(x)dx$$

Where *f(x)* is the probability density function of frame times from the participant nodes. So,

$$P(X \leq W) = \int_0^W \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}} dx$$

$$= \frac{1}{2}\left(\mathbf{1} + \mathbf{erf}\left(\frac{W-\mu}{\sigma\sqrt{2}}\right)\right) \tag{1.1}$$

Since all participants share the download bandwidth of MCU, the end-to-end delays will increase with n, as $d_i \propto \frac{n}{B_{MCU}}$. Similarly, the waiting time in the buffer will also rise with *n*, as $b_i \propto \frac{n}{C_{MCU}}$. Here, B_{MCU} is the total bandwidth and C_{MCU} is the computing capacity of MCU. So, the frame times, i.e., mean delay μ increases with *n*. Therefore, according to equation (**1**.**1**), as *n* increases, the probability that a participant's frame will be included in the composite stream decreases under a given frame expiration time *W*. In other words, given a fixed small *W*, if the number of participants can grow to very large, many frames will be dropped because the frames will take a long time to be delivered for composition. Figure **1**.**4**(b) – (d) explain the concept where the x-axis shows time up to the frame interval. The y-axis represents the frame time. Each bar in the chart is a frame time for a node. The red bars represent the larger frame time than W. Therefore, the corresponding frame is dropped. Given fixed W, the percentage of red bars are higher when *n* is higher during a composition, so more frames will be dropped. If *n* is very large and the composite frame interval T is small, then many participants will not be able to deliver frames on time (because of limited bandwidth) at each composition. Moreover, if W and *n* both are large and W>T, then the outdated frames will be included in the compositions. It is also possible that the MCU will not be able to finish processing all the *n* frames before T expires even they are available because of limited computational power. Both the *T* and *W* are related to human visual perception, and they are usually small to ensure the desired quality of service(QoS). Thus, satisfying the temporal requirement in central MCU based MTS is extremely challenging for large systems like CMTS.

1.4.4 Overcrowding Challenge

Suppose the CMTS with the central MCU can accommodate maximum n participants. So, all the participants can deliver their streams within the given frame expiration time W, and the MCU can maintain the desired frame rate composing all participant streams. If the number of participants further increased, say $n + 5$, the frame time for all participants increase. Therefore, one or more participants will be dropped in the composite frame. We refer to the situation as overcrowding. It degrades the overall quality of service. A scalable CMTS must accommodate a large number of participants, even beyond the system's capacity but still ensure the desired quality of service. Handling such overcrowding situations in CMTS would be challenging.

1.5 A Peer to Peer(P2P) approach for CMTS

A P2P system is a distributed network architecture where the participants share their hardware resources (processing power, storage capacity, network link capacity, etc.)[15]. So, in P2P MTS, the participant devices collectively can perform all the MCU operations, using their shared resources. Such a system can be preferable as it does not need any expensive servers, and no service providers are involved. It also allows users to keep full control over their security and privacy. The decentralized nature of P2P networks increases robustness because it removes the single point of failure, which is a common issue for the central MCU-based system. However, in a self-organizing P2P system, the peers are autonomous, i.e., they can come and go anytime they want. But different mechanisms for incentivizing resource sharing can be used to improve the stability of the system[32]. Another major challenge of using the P2P network is that the peers are vulnerable to many security threats because they need to work as both client and the server. So, direct P2P communications are usually restricted by network address translators (NATs), firewalls, and other network barriers. Interactive Connectivity Establishment (ICE) is a technique used in computer networking to find ways for two computers to talk to each other as directly as possible in peer-to-peer networking [33].

The P2P system has been observed for large systems, including Gnutella[34, 35], Bittorent[36, 37], and Bitcoin[38]. Recently, the peer to peer implementation of video conferencing systems has become popular because of the WebRTC framework, which allows video communication from browsers, i.e., no need to install any specific software packages[39]. But for CMTS, a pure peer to peer mesh network is not suitable. The WebRTC based implementation of MTS can use central plugins (running on one or more peer computers) as SFU or MCU. It is shown that the MCU is better than SFU in terms of CPU and bandwidth consumption [20, 21], thus the system's scalability. However, a central MCU unit in P2P WebRTC implementation is still a bottleneck for the CMTS. Moreover, the P2P network is dynamic, where the autonomous participant devices frequently join and leave the network. So, the research question is, *"Considering the challenges of CMTS and the existing issues of the P2P system, can we implement the CMTS on the P2P network by distributing the operational load of MCU among the participant peers?"*

1.6 Key Terminology

Peer: A peer is a device that a participant(s) uses for telepresence.

Stream: A continuous flow of multi-channel or multisensory (audio, video, light, temperature, air) data over the network.

Conference rate: In P2P MTS, the **conference rate** is the bit rate at which the streams are conveyed between peers.

Video Frame: A frame or a video frame is one of the many still images which compose the complete moving picture or video.

Boxing or mixing: It is the process of scaling and combining multiple streams into one stream of the conference rate. Video mixing is also referred to as "video composition" throughout the dissertation.

B**oxing capacity:** The boxing capacity of a peer is the number of streams of a specific rate the peer can box.

Profile Weight*:* it can be calculated from public profiles of a participant such as Google Scholar, ResearchGate, Facebook, etc., or the participant role in a particular MTS session.

Demand Score*:* It is the number of requests for the stream of a participant from other participants

Stream Score: It is the score assigned to the stream of each participant. In this dissertation, it is calculated as the multiplication of profile weight and the demand score.

DRDI: It stands for Dynamic Role and Demand based Index. Suppose in an MTS session; each participant i has a ***profile weight*** w_i and a ***demand score*** v_i. So, the ***stream score*** of the participant i can be defined as $p_i = w_i v_i$. Then, a simple **DRDI** can be defined as $\sum_{i=1}^{n} p_i$, where *n* is the number of participants. The maximization of DRDI within a timing constraint can enable the network to prioritize and carry only the streams of essential participants and block the less critical streams. It will help to handle the overcrowding in the MTS.

SCDT: It stands for the Stream Collection and Distribution Tree. It can be any tree generated from an undirected connected graph. The name suggests that it can be used as a communication network for the CMTS. In other words, it can be used as an application-layer multicast(ALM) tree where the participants can forward their streams towards the root of the SCDT. The root can generate the final composite stream and returns it to the participants using the same ALM tree.

Waiting Time: As discussed earlier**,** the frame interval *T* and the frame expiration time *W* are related to the human visual perception. Usually, they are small to ensure the desired quality of service(QoS). For simplicity, let us consider W=T. Then, the MCU waits for the frame interval time W or T to collect streams. We refer to this interval as the **waiting time**. The frames that are generated within the given waiting time are inserted in the composite stream and discarded otherwise.

1.7 Dissertation objectives

The dissertation aims to answer the research question mentioned above. It presents a peer to peer architecture of CMTS based on the Principle of Distributed Computing (PDC) and the Principle of Priority-based Resource Allocation(PPRA). These principles are used to address three of the four challenges discussed earlier (1) Computational Challenge, (2) Temporal Challenge, and (3) Overcrowding Challenge. The PDC addresses the computational and temporal challenges by allowing the distribution of the MCU workloads among participating devices. The PPRA targets the temporal and overcrowding challenges of CMTS. It helps to utilize the limited resources of the P2P network properly. It is used to design a profit-based stream collection mechanism for maximizing a Dynamic Role and Demand based Index (DRDI) in each stream composition performed, satisfying the timing requirements and network resource constraints. This dissertation does not address the 'Visual and Cognitive Challenge' challenge because it is a different research dimension. Major studies and considerations are necessary as different applications of MTS need different visual layouts. The grid-view layouts are currently used in many commercial MTS systems. In [40], a 2D gallery-like layout is presented for designing Digital Amphitheater (DA), where the performer or panel members are shown in a different section. A design 3D tele-immersive(3DTI) Amphitheater is discussed in [41], where the virtual stage is constructed using a 3D model of performers generated using 3D streams from their physical performing location. Based on the layout and the type of streams, the computational and communication load will be different. A general research question can be, “can we support large-scale visualization layouts for different applications optimizing the resource consumption of the system?” It also opens many sub-questions for research because content-specific designs of layouts or environments are required for volumetric video construction, virtual reality, augmented reality, etc.

1.8 Problem definition

For a P2P implementation of CMTS, MCU's operational load must be distributed among the participating peers. For that, a peer must be selected to work as MC and some other peers as MPs. The MC peer needs to communicate with MPs for task distribution and the final composition of streams. So, it is crucial to optimally place the MC and MPs in the network, considering the computational and communication capacities of the peers. In CMTS, overcrowding is expected, but the system still should generate a composite stream at a given bitrate, i.e., the frame rate. If some streams are required to drop from the composition, the system must drop the less essential streams.

Suppose there are *n* participant peers in a CMTS where $\mathrm{n} \geq 2$. Their logical topology forms an undirected graph G = (V, E), where *V* is the set of all participants, and *E* is the set of edges among them. Each participant node *i* has upload bandwidth U_i, download bandwidth D_i, video mixing capacity B_i. The participant also has a ***profile weight*** in the conference w_i(can be calculated from public profiles such as google scholar, Facebook account, etc. or the role of the participant in the session) and a ***demand score*** v_i which is the number of requests for the stream of *i* from other participants. The individual

stream score of participant *i* can be defined as $p_i = w_i v_i$. Suppose during a CMTS session, the system waits for a given waiting time *W* to collect streams from the participants, then the MC and MPs collectively combine the streams to generate the composite stream. So, the waiting time *W* defines the frame rate or bitrate of the composite stream. We must select the MC and MP nodes among all the participants peer so that the following profit or DRDI function is maximized under the waiting time W,

$$\boldsymbol{P} = \int_{\boldsymbol{0}}^{\boldsymbol{W}} \boldsymbol{C}(\boldsymbol{t})\boldsymbol{dt} \qquad (1.2)$$

Where $C(t)$ is the accumulated profit by the system over time *t* and it can be defined in terms of all parameters of the participant peers.

1.9 Solution Approach

The problem asks to find an optimal solution in the exponential solution space (MPs can be chosen in 2^{n-1} ways after picking an MC node) satisfying multiple constraints. It would be too expensive to find an optimal solution for a large-scale system like CMTS. So, we consider building an SCDT first from the given graph G, keeping the MC as the root. Then some MPs are selected around the MC node. According to the definition of SCDT, in a CMTS session, each participant originates a raw multi-channel stream of conference rate *s*, say only if $v_i > 0$. These streams travel towards the MC using the SCDT. The SCDT has the MPs that partially mix all the streams traveling towards the MC and forward a single mixed stream. So, the MPs help the MC to generate the final composite stream. The MC node waits to collect streams for the next composition until the given waiting time W is expired. The goal is to include all participants so that the profit function is maximized. Since each MP node j performs the partial mixing, it also needs to wait for a time T_j so that it can maximize the profit in its partial mixing. For the non-MP nodes, the waiting time is 0, i.e., they receive and forward. The MC node returns the composite stream to the participants according to their interests. However, the dilemma arises to set waiting time T_j for MP nodes. Because the small value will cause to miss many streams from the descendants, and too high value will cause unnecessary delay or even miss timer deadline of the next ancestor MP in the SCDT. So, considering the waiting time W given for the MC, we must build the SCDT by optimally placing MC and MPs and assigning the waiting time T_j for each MP j. The process of building SCDT must maximize the profit function defined in (1.2).

1.10 Contribution

The SCDT formation problem discussed above still asks to find solutions in the exponential search space (MPs can be chosen in 2^{n-1} ways) satisfying multiple constraints. It would be too expensive to find an optimal SCDT for a large-scale system like CMTS. So, in this research, a constructive phased design approach is explored that divides the search into multiple phases by addressing **different profiles of the profit or DRDI** function. The approach generates the near-optimum solution, which is shown later.

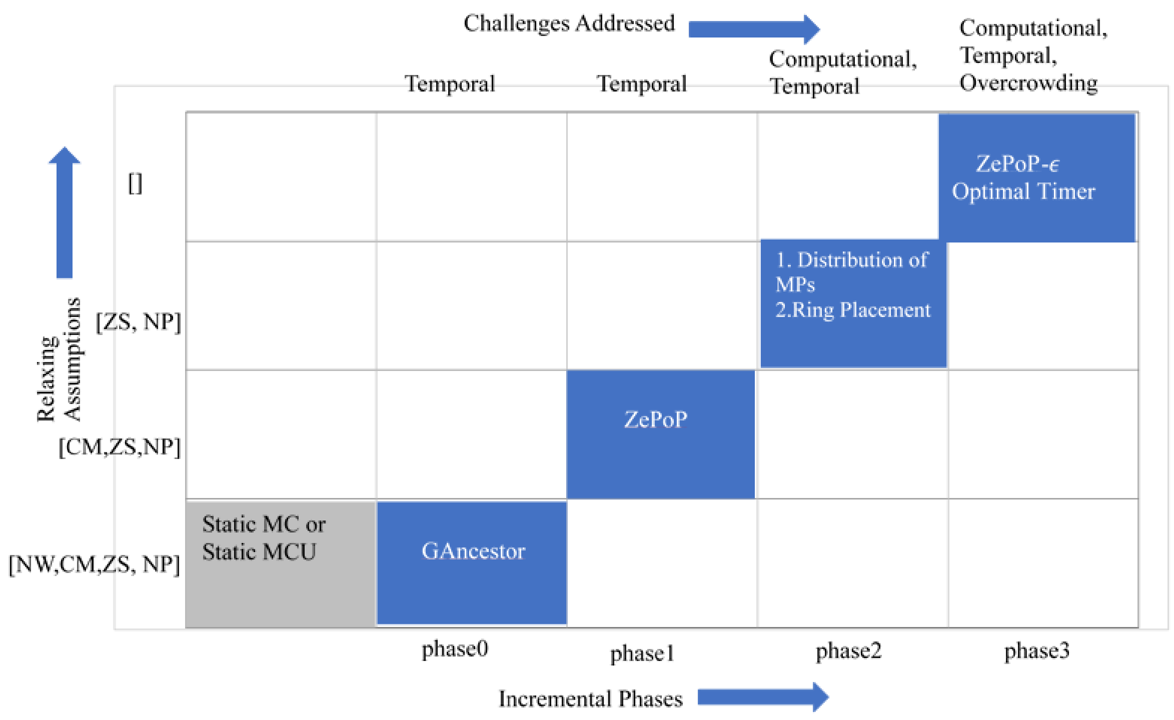


Figure 1.5 Summary of incremental design of the solution

1.10.1 Incremental Design Approach

Given the complexity of the problem, the dissertation has adopted a phased design approach. The design starts with the general communication architecture of Gnutella for P2P protocol[42] and adopts it for a simple scenario based on some assumptions. The dissertation then, one by one, relaxes those assumptions to create the full solution. In the phase0, a simple version of the problem is considered to solve. The phase is designed based on four assumptions. These are,

1. No Weighted Links(NW): The hop count on a communication path is equivalent to the path delay.

2. Centralized MCU(CM): Each node has enough bandwidth to receive and process streams from all the participants so that only one MC and MP are needed for the CMTS.
3. Zero Sync(ZS): no synchronization is required among the processing units, MPs
4. No stream priority(NP): The stream score is one for all the participants

In subsequent phases, different combinations of these assumptions are relaxed, addressing various challenges of CMTS. Figure **1.5** shows a summary of the incremental design of the entire solution. The entries of the table are protocols designed in different phases. The protocols in the final phase relax or remove all the assumptions.

1.10.2 Phase0

So, in phase0, with all the assumptions applied, the problem turns into the optimal placement of one MC in the P2P overlay network that maximizes a profit function. So, in this phase, a complete protocol called **GAncestor** is presented for the MC placement in a dynamic P2P network. The GAncestor is, however, a complete peer-to-peer protocol i.e., it addresses the autonomous node-join and departure. It can form and maintain the optimal SCDT by maximizing a profit or objective function. It defines the message scheme as well as multiple algorithms. The algorithms are validated using theoretical proof as well as experiments. In the experiments, a single MC and MP based CMTS is emulated where the SCDT maintained by GAncestor is used for stream communication. As the protocol selects a unique peer as the MC among all the peers, it can also be used as the leader election protocol in autonomous distributed systems. The GAncestor protocol is discussed in CHAPTER 3. The algorithms of GAncestor are used in the next phases or chapters.

1.10.3 *Phase1*

In this phase, a generalized version of the protocol **GAncestor** is presented by relaxing assumption 1. Therefore, the weighted links are considered for the MC election. The protocol is renamed to **ZePoP** and discussed as a leader election protocol for a dynamic P2P environment. The leader election algorithm, which is the main part of ZePoP, elects one of the peers as the leader, maximizing a profit or objective function defined in terms of delay-based closeness centrality. So, it needs to consider actual weighted links for defining and optimizing the profit function. The single MC-based telepresence system (as discussed in phase0) is shown as an application where the leader works as the MC sitting at the root of the SCDT. ZePoP protocol is discussed in CHAPTER 4. It is used in the next phases to pick the MC and distribute the MPs around it.

1.10.4 *Phase2*

In this phase, assumptions 2 and 3 are relaxed by forming the distributed MCU. So, after placing the MC using either GAncestor or ZePoP, one or more MPs need to be selected in the SCDT. For that, two different methods of placing the MPs around the MC are discussed. In the first method, MPs are placed satisfying the constraints on bandwidth and computing capacity of the peers. The MPs are placed around the optimally placed MC (using GAncestor) to increase the accumulated profit or the DRDI further. A waiting time management scheme is also discussed to set the waiting time T_j at each MP node j. However, in this method, assumption 3 is kept in place, i.e., no synchronization cost among the MPs are considered. In the second method, a ring placement analytical model is discussed where MPs are picked at a fixed distance from the MC in the SCDT considering the synchronization cost among the MPs; therefore, assumption 3 is removed. The phase2 is presented in CHAPTER 5.

1.10.5 *Phase3*

In this phase, all three assumptions are relaxed. This phase aims at enhancing the scalability of the system by addressing the overcrowding challenges. An extended protocol of ZePoP called $ZePoP\text{-}\epsilon$ is discussed for the MC placement. It includes the stream score of the participant nodes in the profit function of the system. The protocol ensures that if the stream dropping is necessary because of overcrowding, then the streams with low scores are dropped. The MPs are selected, satisfying the constraints using phase2. Later, an optimal timer assignment technique is discussed that aims to maximize the DRDI further within the given waiting time at the MC. This final phase finds the near-optimal solution of SCDT based on an extended profile of the DRDI or profit function. CHAPTER 6 discusses this final phase.

1.11 Dissertation Outline

In CHAPTER 2, related works of different parts of this dissertation are discussed. In CHAPTER 3, a complete protocol called **GAncestor** is presented for the MC election in a dynamic P2P network. It includes the necessary message scheme, algorithms for node joining, departure, and the MC placement. The protocol addresses the phase0 by hop-count based minimization of the mean delay from all nodes to the MC. A generalized version of the GAncestor protocol called ZePoP is presented in CHAPTER 4, which places the MC based on a delay-based metric such as closeness centrality. So, CHAPTER 4 addresses the phase1. Both versions of the protocol aim to create an optimal SCDT but do not consider the participants' stream scores. In CHAPTER 5, the distributed MCU is formed to relax assumptions 2 and 3. For that, a couple of methods for distributing MPs on the SCDT are discussed as the parts of Phase2. The first method aims to place the MPs satisfying the constraints on the resources of the peers. For the second method, an analytical model is presented, which suggests putting a ring of MPs on the SCDT without worrying about satisfying the node constraints. An adaptive waiting time management scheme is also shown in this chapter that is used to assign timers at MPs. CHAPTER 6 addresses the final phase or Phase3, relaxing all the assumptions of this dissertation and presents the final role-based design of the profit function of the CMTS. The MC placement protocol $ZePoP\text{-}\epsilon$ includes the scores of the individual streams in the objective function. Then an optimal timer scheme is presented that can assign the optimal value of waiting time at MP nodes for maximizing the overall DRDI of the system. CHAPTER 7 concludes the dissertation with some future research directions.

CHAPTER 2

Related Work

2.1 Introduction

For large-scale video conferencing, various public and private video conferencing providers such as Zoom, Google Hangouts, WebEx, Polycom, etc., make one or more MCUs available in multiple parts of the public network. Now, instead of placing dedicated hardware MCUs, cloud-based software MCUs are becoming popular to address the scalability issues of video conferencing [43]. However, these cloud servers are still expensive for large-scale systems. So, as we investigate the alternative P2P solution. This chapter discusses the related works in the context of the contribution of the dissertation.

2.2 P2P Video Conferencing

As alternatives of systems with dedicated MCU server(s), several notable attempts have been made towards realizing MTS on distributed autonomous peer-to-peer (P2P) service model. In the P2P schemes, all participants exchange their streams without the need for any service provider managed MCUs. However, such unmanaged architecture generates excessive network traffic. Various extensions were proposed to design and implement workable systems. Some of them focused on the scalability reigning excessive traffic. Civanlar et al. (2005), [45] showed an extreme schema where video conferencing can be performed among deficient bandwidth participants. But the schema would restrict a peer to send one and receive only one video stream at a time, which caused to deny video-request from a new participant in some cases, even with a small number of participants. In [46], Akkuş et al. (2011) extend the approach in [45] with scalable video coding techniques and dynamic rewiring of peers to further maximize participants in the speaker-only MTS. Anh Le and H. Nguyen (2010) [17] present a comparative analysis between MCU-based and distributed P2P conferencing. In P2P architecture, the peers share video streams using the Application Layer Multicast (ALM) tree [16]. They show that a highly expensive MCU is needed to match performance with the P2P systems in terms of quality and end-to-end delay. In [47], a distributed architecture of a video conferencing system based on P2P systems is presented where some cooperative peers form an overlay network of distributed media control servers(MCS). The MCSs mainly help to build ALM Tree for different sources considering all video requests. Xuan Zhang et al. (2013), [48] discusses a hybrid architecture where the participants directly send video stream to a single node called "reflector," then the reflector distributes video streams to all participant using an ALM tree. Thus, they claim load (computational and bandwidth) reduction on reflector compared to MCU-load in a star network. S. Petrangeli et al. publish a series of works to show the scalability of WebRTC based P2P system [49-51]. They use a star topology centered at a conference controller (a SFU) that helps to improve quality of service by dynamic bitrate calculation and stream selection, improve scalability by limiting number of encodes at senders. In this research, the issues of single point of failure and the scalability issues of P2P approaches are futher addressed by distributing the MCU load among the participating peers, and prioritizing the streams to be carried through the network.

2.3 Election Protocols for MC selection

In a P2P system, the peer nodes can come and go anytime. In the proposed P2P solution of CMTS, it is required to dynamically select one of the peers as MC in response to the dynamic nature of the network. Many works have been offered for leader election in distributed systems. One of these algorithms can be used to design an election protocol for finding the MC peer, but we need to place the MC optimally and maximize the DRDI defined in the previous chapter.

So, the election protocol must minimize the average path distance to the MC from all other nodes, or equivalently maximize the closeness centrality. In general, the closeness centrality C_x of node x is defined as shown below [52],

$$\boldsymbol{C_x = \frac{n-1}{\sum_{y=1}^{n} D_{yx}}} \qquad (2.1)$$

Where n is the total number of nodes in the network, and D_{yx} is the distance from the node y to x. Some earlier papers present the leader election protocols based on the known logical topologies of the systems such as ring, complete graph, tree, etc.[53-56]. However, the position-based leader election must consider the end to end distance in the real network topology, which can be arbitrary in structure. Mega-Merger and Yo-yo[57] are among some universal leader election algorithms that work for arbitrary topology. But they elect the leader based on the unique identifiers of the nodes (i.e., the largest or smallest ID holder is the leader) or their randomly proposed numbers. Many attempts have been made to merely estimate the closeness centrality using the neighborhood information of the nodes[58-60]. But only a few works have used the centrality measure in leader election, and a handful of them have used distributed algorithms.

In [61], W. Mary et al. presents a central algorithm for leader election using closeness centrality calculated from the adjacent matrix. K. ChongGunM and W. Mary present a distributed leader election mechanism with a tree-based centrality measure[62]. At first, they form a tree-connectivity among the nodes rooted at a random initiator of the process. In the next phase, the initiator collects the layer and depth information of each node in the tree, calculates the centrality for them, and selects the node with the highest centrality as the leader. In the third phase, the same tree is used to announce the leader. They try to reduce the messages for the election but takes three phases. Moreover, the result highly depends on the initiator that start creating the tree. K. Mokhtarian and H. Jacobsen [63] discuss the algorithms for forming a minimum delay overlay multicast tree. The aim was to allow any node to build the tree if it requires to deliver delay-sensitive data to a group of receivers with minimum delay. They do not consider the leader election, but the root of the tree can be considered as the optimal leader in terms of delay.

N. Andre et al. recently have worked on several centrality-based leader election algorithms[64]. They state that “selecting a central node as the leader can significantly improve algorithm efficiency by reducing the network traffic or the time of convergence, especially in Large-scale lattice-based Modular Robots (LMRs) that form large-average-distance and large-diameter networks. In time-master-based synchronization protocols, placing the time-master at a central node leads to more synchronization precision in large-diameter networks as the precision of remote clock readings tends to decrease with the hop distance". Their ABC-Approximate-Center Election algorithm is like a leader election with hop-based closeness centrality[65]. They also propose the k-BFS SumSweep algorithm designed to elect an approximate center node[66]. Both algorithms are specially designed for choosing the center node as a leader in LMRs. They also worked on approximating the network centroid for large scale Embedded Systems[67]. For that, they use effective closeness centrality presented in[68] and the tree-based leader election mechanism mentioned in[62].

2.4 Distributed MCU

In P2P architectures, streams are processed in a distributed manner in most cases. As the previous section already discussed the related work based on P2P design, this section discusses the distributed MCU architecture based on dedicated cloud servers or devices. In [69], G. Boris et al. consider using geo-location of the participants to select cloud servers to form on-demand distributed MCU. They pick the servers as the participants join or leave, optimizing the cost as well as the Round-Trip Time(RTT) between the participants. However, the approach connects all participants from a region with a single regional server (star network), which can be a bottleneck for large scale local conferences. The single server hosting the MCU would be expensive as well.

Rodríguez et al. (2016) present a room-based multiparty video conferencing scheme with distributed cloud-based software MCUs. They have published a series of research articles starting from design to the real implementation, presenting different metrics for system evaluation [7, 70-72]. The MCU is divided into multiple stream broadcasters called OneToMany (OTM) processors so that they can be hosted on different cloud servers. Each OTM receives a video from one participant and broadcasts it to all other participants in the room. So, in a room of N participants, there are N OTMs in the MCU server(s), N incoming links to the server(s) and N^2 outgoing connections from the server(s). So, the system is not only CPU intensive but also requires excessive bandwidth at the servers hosting OTMs. The authors evaluate the system based on CPU usage but skipped analysis on bandwidth consumptions, which is very important to the scalability. In summary, the large-scale implementation of the system would be costly. The author suggested to study the financial model as future work, but nothing has been done yet.

CHAPTER 3

GAncestor Protocol

3.1 Introduction

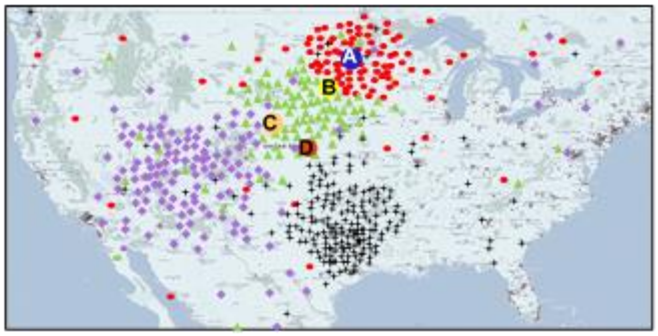


Figure 3.1 Movement of MC in CMTS. Node joining order with shapes: circle→triangle→diamond→plus and MC movement with letters: A→B→C→D

In this chapter, a complete peer-to-peer(P2P) protocol called **GAncestor** is presented for the MC election in a P2P CMTS. The protocol solves the simplest version of the optimization problem according to the phase0. However, it is a complete peer-to-peer solution that can handle autonomous node ID allocation, link maintenance, node join, and node departure tasks. The protocol builds an optimal SCDT keeping the MC as the root. An optimal or balanced SCDT can significantly reduce bandwidth usage and end-to-end delay in a telepresence session. It can help to scale up the overall system and thus maximize the DRDI. So, the CMTS is assumed to be run on a balanced SCDT rooted at the MC, and it has only one MP, which is the same as the MC.

3.2 GAncestor Protocol

In the P2P system, the peers frequently come and go, i.e., the topology changes. For adapting to the dynamic nature, the MC node must be dynamically migrated among the peers to keep the SCDT always balanced. For example, Figure 3.1 shows a scenario of CMTS where a different swarm of peers joins an event using audio-video streams at different times (displayed with different shapes). The MC corresponding to the network dynamically moves (shown with letters A, B, C, D) to minimize overall conferencing delay and bandwidth usage by reducing the total or average hop-count from all participants to the MC. This way, the SCDT rooted in the MC will be balanced in terms of hop count in its branches. On this basis, the GAncestor protocol aims to form the SCDT by minimizing the total or average hop-count to the MC, i.e., maximizes the DRDI.

The proposed protocol has all the essential components that are required for any protocol on the P2P system. It has a complete message scheme for peer to peer communication and algorithms for handling node joining, node departure, node failure, etc. Some of these algorithms are based on the popular P2P system Gnutella [35]. The protocol uses an objective function to achieve the optimality of the SCDT. Each node in the network runs a distributed MC election algorithm, designed for P2P CMTS, to create the SCDT optimizing the objective function. The following subsections discuss different components of the proposed protocol.

3.2.1 The Objective Function

For the P2P CMTS, the objective function is the profit function, as defined in equation (1.2). If the MC to be selected is the node M and the given waiting time W, then it can be rewritten as,

$$\boldsymbol{P_M(W) = \int_0^W C(t)dt} \tag{3.1}$$

Where $P_M(W)$ is the profit accumulation at MC node M within a given waiting time W. The profit accumulation from a single node *s* to the MC can be defined as the multiplication of the stream score of node *s* and the probability that it will reach MC within the given waiting time W. Then, the equation (3.1) can be written as follow,

$$P_M(W) = \sum_{s \epsilon V} w_s v_s \int_0^W f_{s,M}(t)dt$$

$$= \boldsymbol{\int_0^W \sum_{s \epsilon V} w_s v_s f_{s,M}(t)dt} \tag{3.2}$$

Where $w_s v_s$ represents the stream score of the node s and $f_{s,M}(t)$ is the probability distribution function of path delays from node s to the MC node M. So, $\int_0^W f_{s,M}(t)dt$ is the probability that the stream of node *s* will be delivered to the MC within the time *W*. Suppose the delay distribution from any node s to the MC node *M* is a normal distribution, i.e., if the delay between s and M is $D_{s,M}$ then $D_{s,M} \approx N(\mu_{s,M}, \sigma_{s,M}^2)$. It is known that the summation of normal random variables follows the normal distribution[73], i.e.,

$$\sum \boldsymbol{w_s v_s X_s} = \boldsymbol{N}\left(\sum \boldsymbol{w_s v_s\, \mu_s}, \sum (\boldsymbol{w_s v_s \sigma_s})^2\right) \tag{3.3}$$

where $X_s \approx N(\mu_s, \sigma_s)$. So, the equation 3.2)can be written as,

$$\boldsymbol{P_M(W)} = \int_{\boldsymbol{0}}^{\boldsymbol{W}} \boldsymbol{f_M(t)dt} \tag{3.4}$$

Where, D_M is the path delay from any node to M and,

$$D_M \approx N(\mu_M, \sigma_M^2), \qquad \mu_M = \sum_{s\epsilon V, s\neq M} w_s v_s \mu_{s,M} \; and \; \sigma_M^2 = \sum_{s\epsilon V, s\neq M} \left(w_s v_s \sigma_{s,M}\right)^2$$

As $D_M > 0$, then mean $\mu_M > 0$. Therefore, $\int_{-\infty}^{0} f_M(t)dt$ is negligible. The equation 3.4) can be written as,

$$P_M(W) = \int_{-\infty}^{W} f_M(t)dt$$

$$= \int_{-\infty}^{W} \frac{1}{\sigma_M \sqrt{2\pi}} e^{-\frac{(t-\mu_M)^2}{2\sigma_M^2}} dt$$

$$= \frac{\boldsymbol{1}}{\boldsymbol{2}}\left[\boldsymbol{1} + \boldsymbol{erf}\left(\frac{\boldsymbol{W - \mu_M}}{\boldsymbol{\sigma_M \sqrt{2}}}\right)\right] \tag{3.5}$$

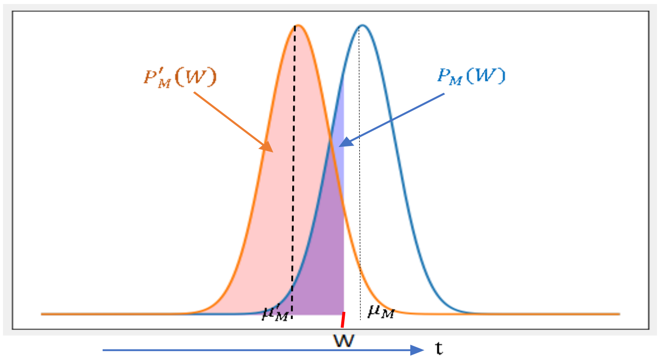


Figure 3.2 The change of DRDI $\boldsymbol{P_M(W)}$ with the change of $\boldsymbol{\mu_M}$under given W

According to equation 3.5), the low values of both u_M and σ_M can maximize the DRDI $P_M(W)$ under a given waiting time, W. Figure **3.2** also shows that if the mean delay (μ_M) from the participant to the MC is reduced (μ'_M) then the profit function or DRDI $P_M(W)$ can be increased ($P'_M(W)$. So, we have to pick a MC node M from all the participants in such a way that the μ_M and σ_M are minimized. According third assumption of the phase0, all participants have the same stream score to the MC i.e., the profile weight $w_s = 1$ and demand score $v_s = 1$ for all participants $s \epsilon V$. Then the mean μ_M can be redefined as,

$$\boldsymbol{\mu_M = \frac{\sum_{s\epsilon V, s\neq M} D_{s,M}}{n-1}} \qquad (3.6)$$

Where *n* is the number of participants in the system and $D_{s,M}$ is the latency from the node *s* to the selected MC node *M*. However, $D_{s,M}$ is also equivalent to the total hop from the node *s* to the Node *M* (*assumption 1*). **A hop count can also be considered as a single unit of traffic for stream communication between two nodes.** So, a new term **total traffic** at the node M can be defined as the total hop counts from all participants the node M, which is,

$$\boldsymbol{H_M = \sum_{s\epsilon V, s\neq M} D_{s,M}} \qquad (3.7)$$

Based on assumption 1, the optimization of H_M will optimize μ_M because $\mu_M \approx \frac{H_M}{n-1}$. Therefore, the DRDI $P_M(W)$ will be optimized. Therefore, equation 3.7) is the objective function of the GAncestor protocol.

3.2.2 Problem Statement

Suppose there are *n* participant peers in a CMTS where $n \geq 2$. Their logical topology forms an undirected graph G = (V, E), where *V* is the set of all participants, and *E* is the set of edges among them. From G, we can build an SCDT considering the root as the MC. During the telepresence session, each peer originates a raw stream of rate *s*, forwards streams of other peers towards MC, and works as MC when required. The MC peer mixes the raw *n* streams into one composite stream, also of rate *s*, and sends it back to all peers. As stream communication is done using the SCDT, we must create the SCDT in a way that the MC minimizes the total traffic or hop-count defined in equation 3.7).

3.2.3 Message Scheme

The P2P network is a logical or overlay network on top of the physical system. The peers in the dynamic overlay network talk to each other by exchanging messages to elect the leader, form and reorganize the SCDT, inform node departure and arrival, etc. During a telepresence session, the peers also need to exchange some messages and direct strings. We assume that each node has a unique ID in the overlay network from S = {0, 1, 2, ..., 2^{10}}. We form and maintain the P2P dynamic overlay network using the mechanism used in the popular, fully distributed system Gnutella[34, 35].

Fields	Node ID
Byte Offset	0 ··· 1

(a) REPLY_ID, ELECTION

Fields	Node ID	Total hop count
Byte Offset	0 ··· 1	2 ··· 2

(b) INFORM, ARRIVAL, DEPART

Fields	Node ID	Node Flags
Byte Offset	0 ··· 1	2 ··· 1026

(c) LEAVE, JOIN, HOTNESS

Figure 3.3 Message formats. "Node ID" refers to the Source ID, but only for REPLY_ID message; it is a New ID for the newly joined node.

The messages used in the system for communication are described below, and their structures are shown in Figure **3**.**3**. The REQUEST_ID and REPLY_ID are used for getting ID for the newly joined node. PING and PONG messages are used to discover some existing nodes in the conference, and their structures are the same as used in Gnutella, except the PONG message has an extra field called "hop count to MCU". The structure of PING is also used for the message REQUEST_ID. The JOIN message is used by the newly joined node to inform MC about its joining. The ELECTION message is broadcasted by every node in the conference to elect a new MC. The LEAVE message is used to inform MC about node departure. ARRIVAL and DEPART are used by the MC to notify all the nodes about a node joining or departure, respectively. A node candidate for MC broadcasts the INFORM message to notify its candidacy. The HOTNESS message is used to inform the parent about the load on a node. The nodes in the CMTS session also exchange some strings (prefix "VCONF") with their direct neighbors, when necessary.

3.2.4 Node Joining Procedure

The node joining steps are almost like Gnutella node joining but need some additional messages to exchange. We assume that the new node can see all existing nodes in the session upon receipt of an invitation from any current node. The node joining steps are described below:

i. The new node initially makes a TCP connection with an existing node in the conference by exchanging “CONNECT” and “OK” string.

ii. The new node obtains an ID by exchanging REQUEST_ID and REPLY_ID message with MC. Then it broadcasts a PING message.

iii. The node receiving PING message responds with PONG message if it wants to accept the new connection, or the PING has the lowest hop count than the previously received one from the same source.

iv. The new node collects some PONG messages, then connects to a random number of the closest PONG responders, but at most α number of them, $0<\alpha<m$.

v. Now, among all the neighbor nodes, the ID of the node closest to the current MC is stored as a direction or gateway towards MC. The lowest node ID is always chosen to break the tie. IDs of other neighbors in the network topology are stored in the list FG_LIST. This list of the future gateway is used to determine the next gateway node in case of the current gateway's departure.

vi. The new node then sends the "SUBS" string to the gateway node to tell that it wants to send and receive streams via that gateway node. The gateway node stores the ID of the "SUBS" string sender in a list SUBS_LIST.

vii. Now the new node sends a JOIN message towards the MC via the gateway node and starts streaming. The MC node receives JOIN messages and decides on initiating the new MC election procedure.

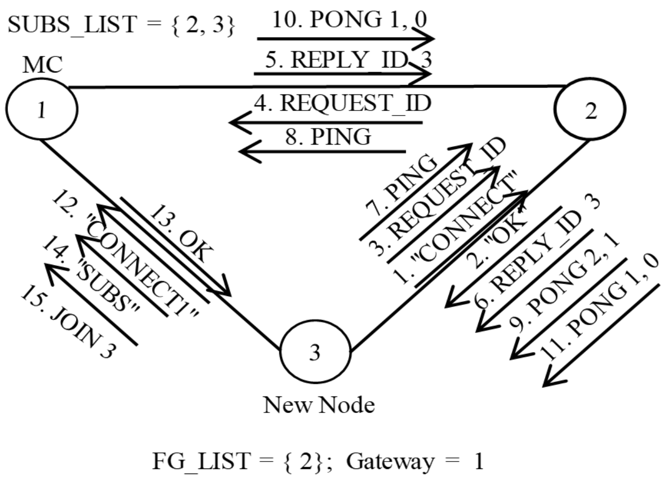


Figure 3.4 Node joining steps. The messages on the links are numbered at the left to represent their order of exchange

An example of node joining is shown in Figure 3.4. The number at the left of each message represents the order message exchange in the joining process. Initially, two nodes of ID 1 and 2 present in the session, and node 1 was the MC. The third node comes and connects to node 2. Then it receives the ID 3 from the current MC using REQUEST and REPLY message. Then PING-PONG messages are exchanged where both node 1 and 2 returns the PONG message, agreeing to accept a new connection. The format of the PONG message shown in the figure is: "PONG Source ID, Hop to MC". After receiving PONG messages, node 3 decides to connect to node 1 as well. When node 1 (current MC) receives the JOIN message, it can determine if it needs to initiate a new election. The decision making for re-election is discussed later.

3.2.5 Link Maintenance and Node Departure

During a telepresence session, each node sends a string "KEEP-ALIVE" to all neighbors after every time interval *t*. The departure of a node is detected by its neighbors upon receipt of the LEAVE message sent by the departed node (graceful leave) or not receiving the KEEP-ALIVE message within time *t* (abrupt leave). For both cases, the gateway node of the departed peer sends a LEAVE message towards the MC. The MC node eventually receives the LEAVE message and decides if re-election is required. If the MC node departs, the neighbors of the MC immediately initiate the election procedure. The node whose gateway node leaves can choose a new gateway node from its FG_LIST or make new connections if the list is empty.

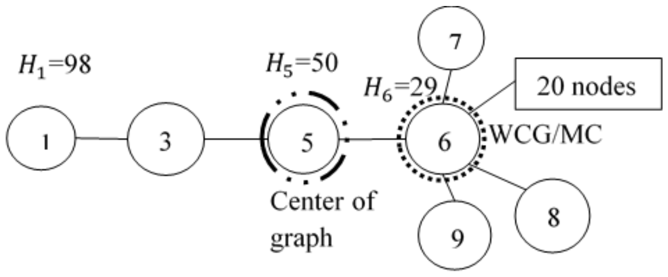


Figure 3.5 Weighted Center of Graph(WCG)

3.2.6 MC Election algorithm

The election algorithm aims to build optimal SCDT. The algorithm picks the Weighted Center of Graph(WCG) as the MC. ***The WCG is the node whose total hop count (defined in equation* 3.7)*) is less or equal to all other nodes in the network. The WCG is better than the Center of the Graph(CG) in terms of total hop-count, as shown in Figure 3.5.*** Thus, picking WCG as the MC, the algorithm makes the SCDT always balanced or optimal. When the election process is started, each node goes through the following steps to decide the MC:

i. If a node receives an ELECTION message for the first time from any node in the session, it forwards the message to others. It participates in the MC election process by broadcasting its ELECTION message.

ii. Suppose two nodes of ID i and j are adjacent. The number of different incoming messages in the shortest path via j to i is denoted by $I_{i,j}$ and the number of different outgoing messages in the shortest route via i to j is denoted by $O_{i,j}$, i.e $I_{i,j} = O_{j,i}$. When the node i receives the first ELECTION message of another node s via neighbor j, then it increases $I_{i,j}$ by 1. If node i has other neighbors such as x, y, z, etc., it forwards the message to them and increases $O_{i,x}$, $O_{i,y}$, $O_{i,z}$ etc. by 1. If the total hop count of node i is H_i and h_s is the hop count of node s to node i, h_s is added with H_i.

iii. If a node i receives an ELECTION message of a node s via a neighbor j, a copy of which already has been received via another neighbor z, then the new message is processed as follows: (a) If the hop count of the new message is less than hop count of the previous message, then $I_{i,j}$ is incremented and $I_{i,z}$ is decremented by 1, and then the message is forwarded to other neighbors, and the number of out-going messages is incremented for them. (b) If the hop count of the new message is equal to the previous one, then node *j* and *z* are given the same preference to receive streams of the message source *s* and that is why $I_{i,j}$ is incremented by *1* and $O_{i,j}$ is decremented by *1* (c) If the hop count of the new message is larger just by one than the previous hop-count, then node *i* and *j* are both equally far from the message source. In this case, $O_{i,j}$ is decremented by 1. If $j = z$, then the new message is forwarded to others only if the hop count of the new message is smaller.

iv. Each node receives at least one ELECTION message from all other *n-1* nodes and then waits for a duration *T* to see if any message with the smallest hop is yet to receive. Then a node *i* decides itself as a candidate for MC if for each of its neighbor *j*, $O_{i,j} > I_{i,j}$. For $O_{i,j} = I_{i,j}$, *min (i, j)* becomes the candidate. Each MC candidate then broadcasts the INFORM message.

v. Each node decides a candidate node as MC whose INFORM message contains the smallest total hop-count considering all INFORM messages received. The lowest node ID is considered to break the tie. At this point, the optimal MC election is done. The remaining steps are to create the SCDT.

vi. Each node sends the "SUBS" string to the neighbor who is closest to the selected MC. Thus, the gateway node towards the MC is chosen. Other neighbors are added to its FG_LIST.

vii. When a node receives a "SUBS" string, it removes the source ID of that string from its FG_LIST and adds that ID to its SUBS_LIST. The SUBS LIST of all peers together is used to create the **SCDT**. The leaf nodes have SUBS LIST empty.

viii. Each node now calculates the number of different streams it has to forward towards the MC during a telepresence session. It is called the **hotness** of a node, which is the number of descendants of the node, and it can be calculated as the total hotness of children plus one. The hotness of the node with empty SUBS_LIST is 1. Each node informs its hotness to its parent using the HOTNESS message. At this point, the SCDT is ready for stream communication.

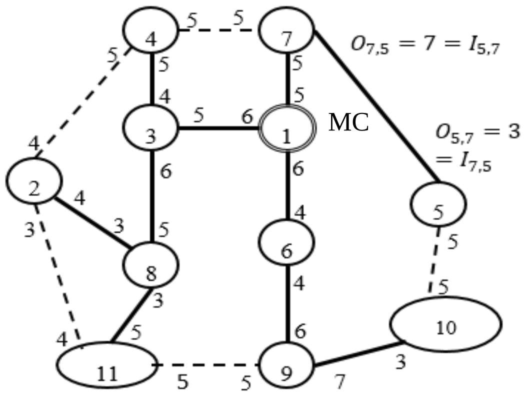


Figure 3.6 An Example topology for MC election

Once the MC is elected, the gateway nodes and the SUB_LIST are used to construct the SCDT**.** The nodes with empty SUBS_LIST are the leaves of the tree. An example of the MC election is shown in Figure 3.6. The numbers on the edges are the number of input and output streams. Recording the shortest paths of ELECTION messages generate these numbers. For example, $O_{7,5} = 7$ means 7 of all ELECTION messages to node 5 will find the shortest path via node 7. At node 1, $O_{1,j} \geq I_{1,j}$ for all neighbor *j*. Therefore, node 1 is an MC candidate. Other candidates are nodes 4, 7, 9, and 11. The total hop count at all of these candidates is 20. So, the lowest ID node 1, the WCG, is elected as MC. The SCDT created by the algorithm ensures the shortest path for each node to the MC, and it is shown in solid lines on the diagram.

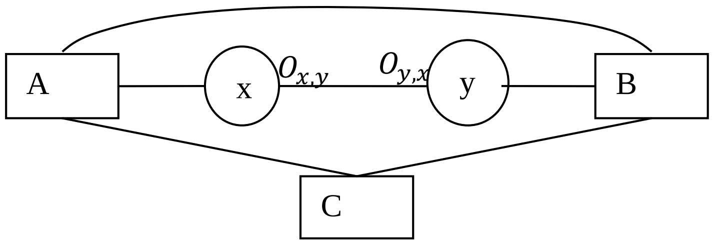


Figure 3.7 Three possible node sets with respect to two neighbor nodes

Theorem 3.1 *The MC election algorithm always elects WCG as the MC ensuring the lowest total hop count.*

Proof: Suppose two nodes *x* and *y* are neighbors in a session's logical topology. Then all other nodes can be grouped in three different sets of nodes, shown in Figure 3.7. The nodes in set *A* are closest to the node *x*, set *B* contains node closest to node *y*, and all nodes in *C* are at equal distance from *x* and *y*. A node in *C* can be connected to *x* and *y* via zero or more nodes but equal in number from set A and B, respectively. A node in set *A* might have another connection to node *y* via at least one node from set *B*. Similarly, a node in set *B* might have another connection to *x* via at least one node of set *A*. So, when all nodes of set A broadcast ELECTION messages, then there is no other best path than via node x to reach node y. Similar case for nodes of set B. Suppose the number of nodes in set *A*, *B* and

C are N_A, N_B and N_C respectively. Then the number of outgoing streams from node *x* to *y*, $O_{x,y} = N_A + 1$ and from *y* to *x*, $O_{y,x} = N_B + 1$. The total hop count of all nodes of set *A* at node *x* is h_A, a total hop count of all nodes in *B* at *y* is h_B and hop count of all nodes of set *C* at both *x* and *y* is h_C. Then total hop count of all nodes to node *x* and y are,

$$\boldsymbol{H_x = h_A + h_C + h_B + N_B + 1} \qquad (3.8)$$

$$\boldsymbol{H_y = h_B + h_C + h_A + N_A + 1} \qquad (3.9)$$

The difference of total hop count, $H_x - H_y = N_B - N_A$

$$\boldsymbol{or, H_x = H_y + O_{y,x} - O_{x,y}} \qquad (3.10)$$

so, if $N_A > N_B$, $i.e\ O_{x,y} > O_{y,x}$ then $H_x < H_y$, hence node x is the better candidate for MC than node *y*. Therefore, we can say, the candidacy of a node depends only on its closeness to other nodes. If there is only one MC candidate *i* in whole telepresence session then it is the WCG and for all of its neighbor *j*, $O_{i,j} > O_{j,i}$. Therefore, in step i*v*, node *i* identifies itself as the MC candidate then broadcasts the INFORM message, and eventually, this only candidate gets elected as MC. According to step i*v*, two neighbor nodes wouldn't be MC candidates. So, if two nodes *p* and *q* decide that they are the MCU candidates, they broadcast INFORM messages. All participants receive total hop count at both *p* and *q* in their INFORM messages. If $H_p = H_q$ then both *p* and *q* are WCG and *min(x, y)* is elected as MC. Otherwise, based on the criteria $H_p < H_q$ or $H_q < H_p$ all participants elect *p* or *q* respectively as MC node (WCG). It works the same way if more than two nodes become

MC candidates. Thus, the algorithm always selects a WCG as MC, ensuring the lowest total hop count at the root of the SCDT.

3.2.7 MC Election Initiation

After an election, the telepresence session continues using the SCDT rooted at MC. The new election becomes necessary when the SCDT becomes highly imbalanced for one or more node joining and departure. The imbalance situation can be detected by one of the following two ways:

(1) *Change of Branch Weight (CBW):* The MC keeps a count of the number of nodes arrival and departure on its every branch. Suppose the MC has *k* branches in SCDT. After the last election, the change (difference of node joining and departure) of number of descendants at *i*th branch is C_i. The current MC initiates election if (max$(C_1, C_2, \cdots, C_k) - \min(C_1, C_2, \cdots, C_k)) > \tau,$ when *k>1;* or $C_1 > \tau$ when k = 1, where, τ is the Imbalance Tolerance Level(ITL) for SCDT.

(2) *MC Candidacy Violation (MCV*): To decide whether to initiate MC election, current MC continuously checks for violation of MC candidacy condition, i.e., the MC node *i* would initiate MC election if for any neighbor *j*, $O_{i,j} \geq I_{i,j}$ becomes false.

The values of C_i or $O_{i,j}$ and $I_{i,j}$ are updated based on receiving JOIN and LEAVE messages. If current MC finds that no election is required based on the above conditions, then it only informs node arrival or departure to all nodes by broadcasting ARRIVAL or DEPART message respectively so that all peers can update their local variables.

3.3 Experimental Results and Discussion

3.3.1 System Implementation

For performance analysis, the CMTS based on the GAncestor is implemented as a distributed application with Message Passing Interface (MPI), C++, and socket programming. The system is emulated on a Cluster of Ohio Super Computer (OSC). The system was tested with 100 participants, i.e., $n = 100$. For creating the P2P overlay networks of different CMTS sessions, it is assumed that each node can connect to maximum m numbers of other participants where the values of m are selected from range [3, 5]. The random values of m are used to simulate the real capability of different machines. α is always set to 2. When the program is run on a computer using MPI on the node with ID 0 (assigned from MPI), other nodes $1, 2, \cdots, 99$ sequentially join the session after every *10* seconds.

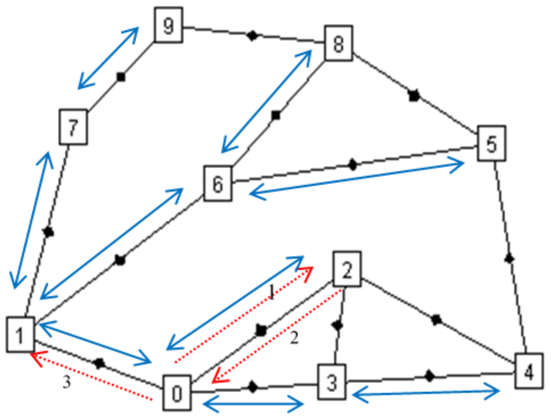


Figure 3.8 A 10-node topology generated by the system

The new node connects through one of the existing nodes selected randomly. As a new node arrives, the current MC decides if re-election is required using CWB or the MCV technique, as discussed in section 3.2.7. After each election, the MC is migrated to balance the SCDT. Therefore, we refer to it as a dynamic MC. Figure 3.8 shows a random topology generated after the first ten nodes join the session. The value of *m* was fixed to 3 for this topology, i.e., the maximum degree of each node would be 3. As described in the algorithm, the PING/PONG strategy of Gnutella is used to discover existing nodes by the new node. But for simplicity, for a new node with ID *t*, an initial node is selected randomly from existing nodes *0, 1,* ···, *(t-1).* The TTL used in the PING message for Figure 3.8 is 3. The dotted red arrows with numbers indicate MC's movement path as nodes 1, 2, 3, ···and 9 sequentially join the session. So, the final MC node is 1 in this case. The bidirectional blue arrows form SCDT, the paths for both raw and composite stream distribution. The randomly generated topology can be of any shape with respect to the initial node 0.

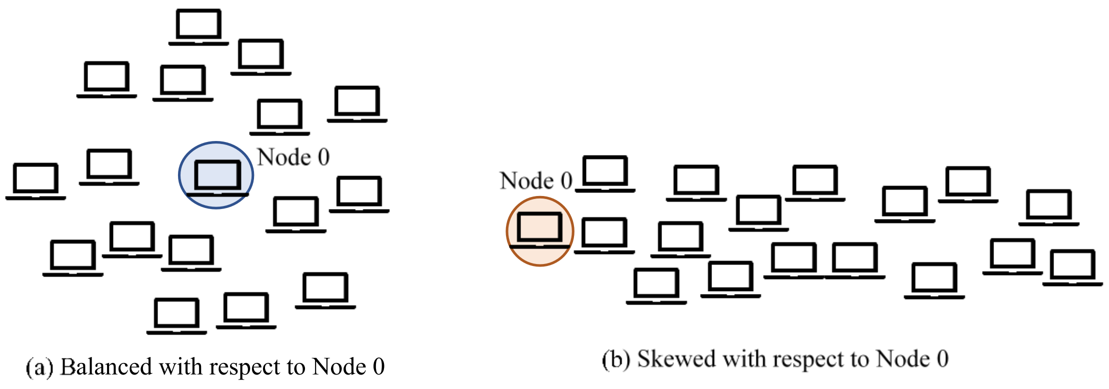

Node 0
(a) Balanced with respect to Node 0
Node 0
(b) Skewed with respect to Node 0

Figure 3.9 Example of topology Structures

A topology can be highly balanced or skewed, as shown in Figure 3.9. Based on the structure of the generated topology, the movement of MC will be different. For skewed topology, MC needs to be moved far away from the initial node 0 compared to the balanced topology to make the SCDT balanced. The performance of GAncester protocol is analyzed for balanced and skewed topologies based on different metrics defined in the next subsection.

3.3.2 Performance Metrics

The performance of the GAncestor protocol is analyzed based on the emulation of CMTS and the metrics considered for analysis are (i) **total traffic**- it is the total hop count from all nodes to the MC node as shown in 3.7). So, the total traffic is equivalent to the mean delay from all nodes to MC. (ii) **Node hotness** – The hotness of a node represents the stream load on the node, i.e., the number of streams the node must forward towards the MC or process during a CMTS session. It can be calculated as the number of descendants of the node in the SCDT plus one. (iii) **Composition time -** it is the time lag between the arrival of the first and last raw stream for each composition at MC. The composition time is the same as the synchronization latency among the streams in the CMTS session. So, the lowest values for all these three metrics are desired from the GAncestor protocol.

3.3.3 Result Comparison

After each election, the application collects total traffic units (total hop count) in the session, hotness of nodes as well as topology information. For the experiment, dummy video streams are transmitted using the SCDT. The size of each packet is size 523 bytes (500 bytes data and 23 bytes header). The MC collects streams from all the participants and generates the composite dummy stream of the same size by taking a portion of everyone's packet. During video stream communication, each node records the jitter to receive packets of the composite stream from the MC. For each packet-level composition, the MC node also records the composition time. The application is also executed, keeping the MC at a fixed location, which is the first node 0. Such an MC is referred to as the static MC. In [48], the static MC is called the ‘reflector’, which collects and distributes the streams from/to the participants using a multicast tree rooted at the reflector. For both static and dynamic MC cases, the single node (MC) process all the streams. In the followings paragraphs, the performance of GAncestor is compared to the system with static MC based on total traffic, node hotness, and composition time.

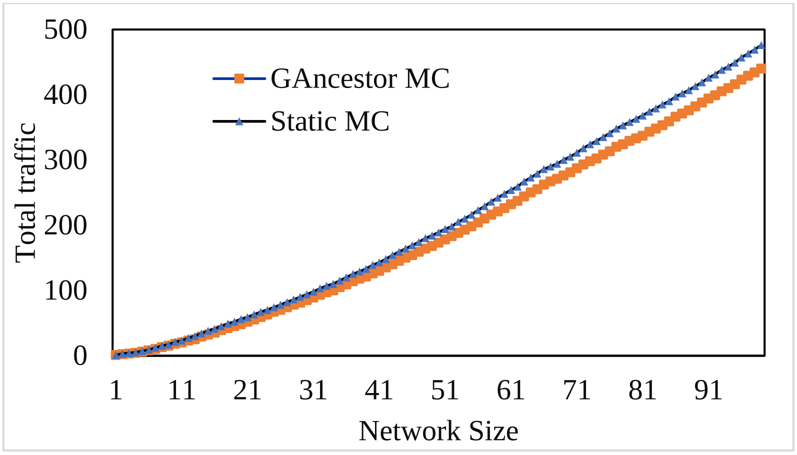


Figure 3.10 Effect on total traffic $\boldsymbol{H_M}$ in the session as the new nodes join.

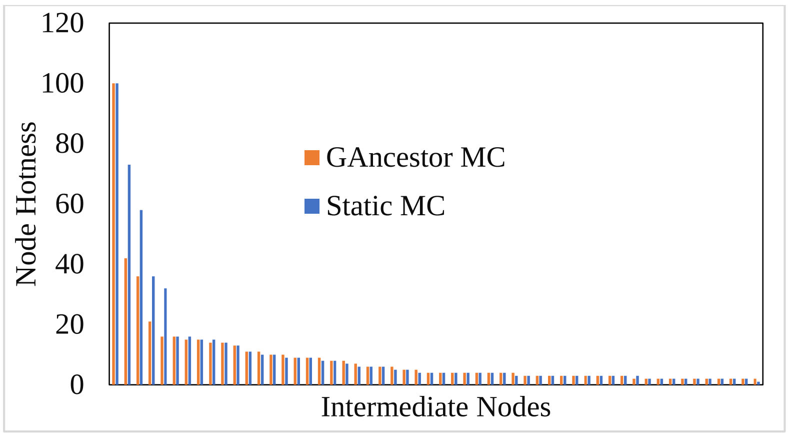


Figure 3.11 Comparison of hotness of intermediate nodes(n=100)

Figure 3.10 shows the comparison of total traffic (average taken over five runs) between the GAncestor MC and Static MC for up to 100-node sessions. As new nodes join the session, i.e., network size increases, the total traffic (hop count) increases in both schemes, which indicates the increased requirement of bandwidth. However, we observe that though the topologies generated by the system were more balanced with respect to the MC node in the Static MC case (node 0), because of the dynamic location change of MC, total traffic is always lower for GAncestor compared to the total traffic in Static MC case. The trend of total traffic increment indicates that the difference will be higher as more nodes join the session. The hotness of a node can be considered as the load on that node. The MC node has the highest hotness as it collects all the streams. Figure 3.11 compares the hotness of intermediate nodes of SCDT for both Static and GAncestor MC cases. The number of participants in the session was 100. The chart shows the hotness values of the intermediate nodes in decreasing order. For both cases, the MC node has the hotness 100 as it collects the streams of all participants. The hotness of intermediate nodes in the Static MC case is significantly higher, especially for the nodes closer to the MC node. These nodes need to forward a large number of streams towards the MC. Given the fixed bandwidth on the links, these overload nodes will significantly increase video delivery time or frame time. The hotness values of intermediate nodes in the GAncestor case are much lower. It will help to reduce the frame time for all the streams. The hotness of the intermediate nodes will be further minimized using load distribution in the next chapters.

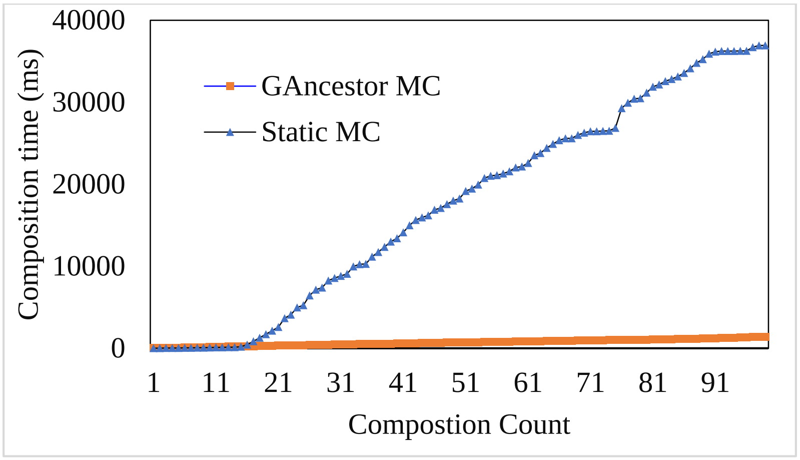


Figure 3.12 Cumulative waiting time to generate composite video(n =100)

For further performance analysis of the proposed approach, we also observed composition time. Figure 3.12 compares the cumulative composition time for 100 consecutive compositions for both GAncestor and Static MC cases. The generated topology for this experiment was highly skewed. i.e., the network grew only one side of the initial static MC node 0, which is possible in many cases of random CMTS sessions. Here we can see, the stream delivery with GAncestor MC is noticeably faster than the Static

MC. We found a similar difference in cumulative waiting time to receive the composite stream at the furthest node from MC. This indicates that the mean delay is very low at a dynamically placed MC compared to the statically placed MC node. Therefore, the DRDI $P_M(W)$ defined in 3.6)3.5) will be significantly high at the dynamically placed MC node selected by GAncestor.

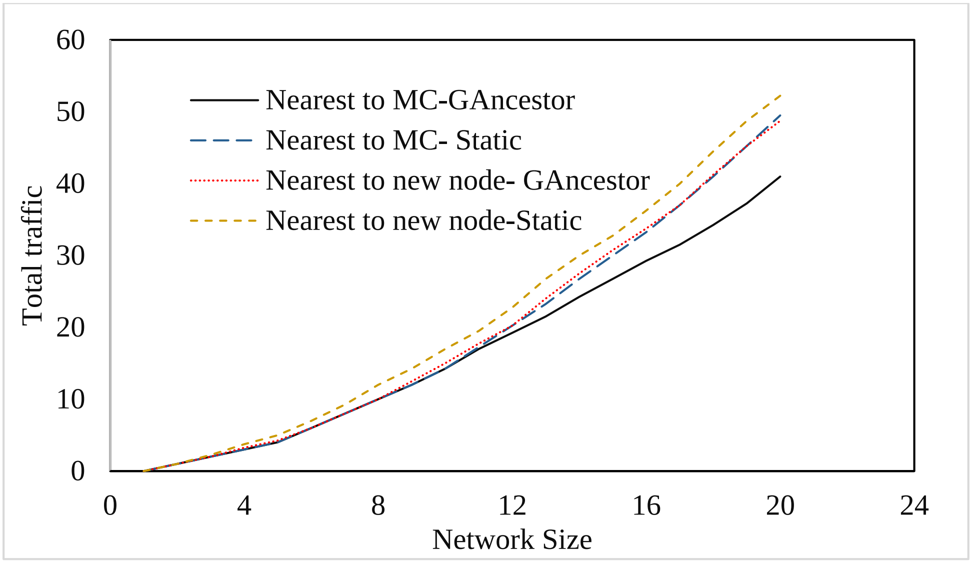


Figure 3.13 Comparison of total traffic when a new node connects nearest to MC or connects nearest to itself.

However, the composition time is calculated based on real latency in the network and highly dependent on the physical location of the nodes participating in the telepresence session. To see a better effect of dynamic MC selection, we must ensure a high correlation between the generated logical topology and physical topology. Though this is another direction of research, a simple approach is considered to check the impact of the location of the initial node for connecting the new node. Two locations are considered for this test, (i) logically nearest to the new node (ii) logically nearest to the MC node. Then, for these cases, the total traffic is observed for both static MC and GAncestor MC. Figure 3.13 shows that connecting the new node nearest to MC and using GAncestor together can significantly reduce total traffic in the system compared to the new node's connection to its closest node (even for the dynamic case). So, the location of the initial contact of the new node is crucial.

3.3.4 Optimizing the Number of MC Elections

Table 3.1 Experimental Results for CBW (with a balanced topology)

Value of τ	T_G	T_S	*TTS (% of T_S)*	*NE*	*NMM*
0	24837	33780	8943 (26.47%)	98	4
2	24840	33780	8940 (26.47%)	17	4
4	24850	33780	**8930** (26.44%)	**9**	**4**
6	24896	33780	**8884** (26.3%)	**7**	**4**
10	24976	33780	8804 (26.06%)	5	3
14	25018	33780	8762 (25.94%)	3	2
20	25334	33780	8446 (25.00%)	3	2

The election algorithm of the GAncestor protocol has a significant impact on minimizing different important network parameters that lead to maximizing the DRDI. However, the election procedure itself takes time. We have observed that the election on 100 nodes takes about 1.2 seconds. If a random delay between 1 to 10 milliseconds is added before sending each ELECTION message (making the nodes slow), the election time becomes almost double. So, the optimum number of elections is desirable to avoid the performance degradation of the system. The unnecessary elections can be avoided by detecting imbalance SCDT by two methods such as CBW and MCV discussed earlier. These two methods are tested on multiple random network topologies of 100 nodes.

Table 3.2 Experimental Results for MCV(Lazy Method)

Example Networks (100 nodes)	**TTS (Eager)**	**TTS (Lazy)**	***% of Eager's TTS from Lazy***	**NMM (Eager)**	**NMM (Lazy)**	**NE (Eager)**	**NE (Lazy)**	**Accuracy of Lazy**
1	3062	3021	98.66%	12	11	99	12	**95%**
2	16570	16569	99.99%	8	7	99	8	**94%**
3	31356	31347	99.97%	12	12	99	13	**96%**
4	4615	4614	99.97%	14	14	99	15	**97%**

Table 3.1 shows the results observed using the technique CBW for different values of τ. The results include aggregate total traffic T_G and T_S for GAncestor and Static MC respectively. The other columns are total traffic saved (TTS) by the GAncestor, the number of elections (NE) performed, and the total number of MC movement (NMM) as the elections are run. The T_G, T_S and TTS are defined as follows:

$$T_G = \sum_{2 \le k \le n} H_M^k$$

$$T_S = \sum_{2 \le k \le n} H_0^k$$

$$\boldsymbol{TTM = T_S - T_G} \qquad (3.11)$$

Where H_0^k is the total traffic or hop count at node 0 (static MC) and H_M^k is the total traffic at the dynamic MC node M when the number of nodes in the system is k. When $\tau = 0$, the SCDT has zero-tolerance to be imbalanced. In this case, TTS, NE, and NMM are all maximum because the election is performed after each node joins. These values decrease as τ increases because the elections are skipped when the new node joins. At $\tau = 0$ the values of TTS and the NMM are ideal, but the value of NE is very high. The best case would be at the lowest value of NE, keeping TTS high and detecting all possible MC movement, i.e., NMM should be maximum. From the table, we can observe that, in a telepresence session of a maximum of 100 participants, the NE would be under 10 to detect all possible movement of MC and minimize traffic significantly given that τ is between 4 and 6. However, the main challenge of this method is deciding a good value of τ for the different number of participants. We find that the MCV method gives similar results compared to CBW(with $\tau = 0$) and has no complexity in deciding any parameter.

Let's redefine the CBW(with $\tau = 0$) as an Eager method because it runs election every time a node joins or leaves. The MCV can be labeled as the Lazy method because it waits to detect an imbalance in the SCDT before running any election process. The accuracy of MCV (Lazy method) can be defined as follows,

$$accuracy_{Lazy} = \frac{1}{2} \times \frac{TTS_{Lazy}}{TTS_{Eager}} \left(\frac{NMM_{Lazy}}{NMM_{Eager}} + \frac{NMM_{Eager}}{NE_{Lazy}} \right) \times 100$$

So, the accuracy depends on how much traffic it can save detecting all the possible movement of the MC. Table 3.2 shows the comparison between MCV(Lazy) and CBW with $\tau = 0$ (Eager method) for four different network topologies of video conferencing of 100 participants. In each run, the MCV(Lazy) method generates TTS and NMM very close (up to 99.99% of TTS) to the ideal values generated by the Eager method. The accuracy of the lazy method is above 95% on average. So, the lazy method(MCV) is preferable because it does not require any parameter settings and yield high accuracy.

3.4 Conclusion

In this chapter, a protocol called GAncestor for the dynamic selection of MC peer is discussed. The protocol dynamically changes MC's location for adapting the dynamic nature of the P2P system and maintaining the balanced branch lengths (in hop count) of the SCDT. The balance SCDT equivalently minimizes the mean delays from all nodes to the MC. Thus, it maximizes the DRDI or the profit function. For simplicity, the profile weights and the demand scores are considered equal (one) for all the participants. The experimental result also validates that the dynamic movement of MC improves performance compared to a statically placed MC in terms of reducing total traffic, individual node hotness, and composition time at MC. Because of the dynamic migration of MC, the P2P approach CMTS also solves the single point failure problem of MC. The overhead related to node maintenance, especially for node departure, is not discussed as it is nearly identical to the ideas presented in many research articles, including[18, 74-76]. As the protocol selects a unique peer as the MC among all the peers, so it can also be used as the leader election protocol in autonomous distributed systems. A generalized version of GAncestor protocol called ZePoP is discussed in CHAPTER 4. It is presented as a leader election protocol for a dynamic P2P network. During a telepresence session, a peer is required to forward streams of other peers towards the MC. In CMTS, this forwarding task can be overwhelming on the nodes at the MC's vicinity, which will slow down the stream composition task at the MC node. The dynamic MC balances this load (the hotness) on the nodes, but for CMTS, it is required to distribute the loads among more nodes, which is discussed in CHAPTER 5.

CHAPTER 4

ZePoP: A Generalized Leader Election Protocol

4.1 Introduction

In this chapter, a generalized version of the GAncestor protocol called ZePoP is presented. It relaxes the first assumption and considers link delays instead of a hop count to form the balanced SCDT. For optimal or balanced SCDT, both GAncestor and ZePoP find a unique node that is selected as the root of the tree. In distributed systems, finding a unique node among all nodes is referred to as the leader election. So, ZePoP is developed as a generalized leader election protocol to be used for dynamic distributed systems where nodes are autonomous. It is designed especially for the P2P applications where the nodes can leave and join anytime, and the leader is responsible for collecting, processing, and redistributing data or control signals satisfying some timing constraints. Thus, it also includes the CMTS, the application under the consideration of this dissertation. The protocol creates a Data Collection and Distribution Tree(DCDT) rooted at the optimally placed leader. The SCDT can be considered as a special version of the DCDT that is used only in the context of stream communication in the CMTS.

4.2 Motivation

Many distributed systems require a leader or coordinator node for system-wide synchronization and making critical decisions. Computing a leader can be thought of as symmetry breaking, where the nodes of the system select one among them as a special node to organize the whole system. In some distributed applications, the leader must collect data from all nodes and redistribute the processed data as fast as possible. For example, in peer to peer video conferencing, the leader needs to work as a Multipoint Control Unit(MCU). It must collect all streams as soon as possible, combine them into one stream and return it to all participants[77, 78]. In the Distributed machine learning on P2P systems, the leader must collect parameters of local models from the computing nodes and generate the global model. During the learning process, the leader also needs to calculate and distribute the model error among peers[79]. The optimal central position of the leader can speed up the tasks in both applications discussed above. Such a leader can also be useful in many Cyber-Physical Systems(CPS) applications such as the data collection and dissemination of control signals in wireless sensor networks, the swarm of robots, or the network of drones IoT, etc.[80].

For the optimal positioning of the leader, the election protocol must minimize the average path distance to the leader from all other nodes, or equivalently maximize the closeness centrality. In general, the closeness centrality C_x of node x is defined as shown below [52],

$$\boldsymbol{C_x = \frac{n-1}{\sum_{y=1}^{n} D_{yx}}} \tag{4.1}$$

Where n is the total number of nodes in the network, and D_{yx} is the distance from the node y to x. Some earlier papers present the leader election algorithm based on the known and fixed logical topologies of the systems such as ring, complete graph, tree, etc.[53-56]. However, the position-based leader election must consider the end to end distance in the real network topology, which can be arbitrary in structure. Mega-Merger and Yo-yo[57] are among some universal leader election algorithms that work for arbitrary topology. But they elect the leader based on the unique identifiers of the nodes (i.e., the largest or smallest ID holder is the leader) or their randomly proposed numbers. Many attempts have been made to merely estimate the closeness centrality using the neighborhood information of the nodes[58-60]. But only a few works have used the centrality measure in leader election, and a handful of them have used distributed algorithms. However, these works are not complete protocols that can be directly used in P2P applications. Moreover, the centrality measures have been considered based on the hop count, i.e., D_{yx} is the total hop count from node y to x. But the hop count-based centrality cannot guarantee the optimal position of the leader in terms of delays. The hop count-based leader election may include a very slow link in the data communication path that increases the overall delay of the system. So, it is crucial to sense the delays of the network dynamically and adjust the position of the leader concerning delay-based centrality measures.

4.3 Problem Statement

Suppose there are n nodes in the dynamic distributed system (the nodes can come and go), where $n \geq 2$. These nodes communicate with each other on a peer to peer partial mesh network or graph. Assume that the graph G = (V, E) represents the communication network where V is the set of all participant nodes, and E is the set of edges among them. One node, called leader, collects data from all other nodes, processes data, and distributes the processed data and control signals. A Data Collection and Distribution Tree(DCDT) rooted at the leader node defines the data path. Creating the DCDT rooted at an optimally placed leader node in G can significantly improve the system performance. So, we have to find a leader node $m\epsilon\ V$ that has the highest closeness centrality in terms of delay. In other words, if the closeness centrality of node *m* is C_m, then to select *m* as the leader, the leader election protocol must ensure $C_m \geq C_u, \forall_{u \in V,\ u \neq m}$. The distance D_{yx} as shown in (4.1) would be the path delay from y to x, which is the summation of point-to-point delays. However, in point-to-point delays, one must consider queuing delay, nodal delay, transmission delay, and propagation delay, especially for the P2P network with heterogeneous devices.

4.4 ZePoP Protocol

The ZePoP protocol defines the mechanisms of leader election for arbitrary P2P systems. We assume that the participant peers form an overlay network using the end to end connection among them. So, the link delay between two peers is the end-to-end delay between them. We also assume that nodes do not leave the network during the election. The election algorithm elects the leader by maximizing the delay-based closeness centrality of the leader. So, the equation (4.1) is the objective function for the ZePoP. The protocol uses other supporting algorithms such as node joining, node departure, and detection of re-election for managing the dynamic properties of the P2P network. The election algorithm works in two phases. The first phase is to calculate the shortest path delays at each node from others as well as record the **branch weight information** for minimizing the number of leader candidates. In the second phase, the recorded branch weight information is used to determine the leader's candidacy and eventually to elect a unique leader. After the leader election, the directional information to the new leader is used to create the DCDT. In the subsequent sections, we discuss the messaging scheme, the algorithms, and proof of correctness.

4.4.1 Message Scheme

The message needed for this protocol is almost the same as the message scheme discussed in CHAPTER 3. Since the purpose is slightly different, the structures are modified accordingly.

Fields	Node ID	d	D
Byte Offset	0 ··· 1	2 ··· 5	6···9

(a) ELECTION, JOIN: the ‘d’ is used to carry point-to-point delay and ‘D’ is for carrying the path delay

Fields	Node ID
Byte Offset	0 ··· 1

(b) REQUEST_ID, REPLY_ID, LEAVE, ARRIVAL, DEPART

Fields	Node ID	Centrality(C)
Byte Offset	0 ··· 1	2 ··· 5

(c) INFORM

Figure 4.1 Message formats. In all messages, "Node ID" refers to the Source ID, but in REPLY_ID message, it is a New ID for the newly joined node.

Figure **4.1** shows the modified structure of the messages used in the protocol. We assume that each node has a unique ID in the overlay network from S = {0, 1, 2, ..., 2^{16}}. The REQUEST_ID and REPLY_ID are used for getting an ID for the newly joined node. PING and PONG messages are used to discover some existing nodes in the system, and their structures are the same as those used in Gnutella. The structure of PING is also used for the message REQUEST_ID. The JOIN message is used by the newly joined node to inform the leader about its joining. The ELECTION message is broadcasted by every node in the system to elect a new leader. The LEAVE message is used for informing the leader about node departure, and ARRIVAL is used by the current leader to inform all nodes about a node joining. A node broadcasts an INFORM message when it decides itself as a candidate for the leader. Besides, the nodes in the system exchange some strings with their direct neighbors when necessary. Strings are sent with the special prefix "VCONF". For example, A new node sends string "CONNECT" as "VCONF CONNECT" to an existing node in the system after making a TCP connection with it. The node accepting the new connection sends back "OK" string as acknowledgment.

4.4.2 The election algorithm: Phase1

Table 4.1 Notations and their initial values

Variables and descriptions	Initial values
d_{xy} $\rightarrow$ link delay between x and y	d_{xy}
D_{sx} $\rightarrow$ path delay from node s to x	∞
NB_x $\rightarrow$ set of neighbors of x in G	NB_x
ϕ_s $\rightarrow$ A node sets this flag to true when it receives the first ELECTION message from the source s.	False
Ψ_{sy} $\rightarrow$ The node x sets this flag if it forwards the ELECTION message of s to the neighbor y.	False
$⋔_{sy}$ $\rightarrow$ Anode sets this flag if it receives the ELECTION message of node s via the neighbor y.	False
O_{xy} $\rightarrow$ the number of ELECTION messages forwarded by x towards the neighbor y in the shortest path.	0
I_{xy} $\rightarrow$ number of ELECTION messages received by x via neighbor y in the shortest path ($I_{xy} = O_{yx}$)	0
G_s $\rightarrow$ the direction of node s	None
FG_LIST → The future parents in DCDT.	{}
CHILD_LIST→ The list of children in DCDT.	{}

ZePoP: Phase1

```
Recording (x, n) // Executed at node x, given the number of nodes in the system.
Begin
  0. message_count = 0; initialize the variables in Table I
Repeat step 1-23 until message_count<n-1
  1.  receive (E, y) // receives Election message via neighbour y
  2.  $D'_{sy} \leftarrow E.D$; s = E. NodeID
  3.  $D'_{sx} \leftarrow D'_{sy} + (E.d + d_{yx})/2$
  4.  if $\phi_s = \boldsymbol{false}$ // receiving a message from s for the first time
  5.      $\phi_s \leftarrow \boldsymbol{true}$;
  6.      Message count += 1]
  7.      $D_{sx} = D'_{sx}$
  8.      Accept (E, s, y)
  9.      Forward (E, s, y)
  10. else if $D'_{sx} < D_{sx}$     // better message has arrived
  11.     resetDirection(s)
  12.     $D_{sx} = D'_{sx}$
  13.     Accept (E, s, y)
  14.     adjustSend (s, y)
  15.     Forward (E, s, y)
  16. else if $D'_{sx} = \boldsymbol{D_{sx}}$ // Equally better message received before
  17.     Accept (E, s, y)
  18.     adjustSend(s, y)
  19. else if $D_{sx} + d_{xy} > D'_{sy}$   // x, y both are in equally better position for s. s in set C
  20.     $D_{sy} = D'_{sy}$
  21.     $T^C_x = T^C_x + D_{sx}$
  22.     $T^C_y = T^C_y + D'_{sy}$
  23.     adjustSend (s, y)
  24. else if s=x // message coming back to x, direct x-> y is slow
  25.     If ($D'_{sy} < d_{xy}$) then $D_{sy} = D'_{sy}$; adjustSend (s, y)
  26. else if s = y and $\boldsymbol{D_{yx} < \infty}$ and $\boldsymbol{D_{yx} < d_{yx}}$ // direct link y->x is slow
  27.     adjustSend (s, y)
End

Note: the variables with prime(') are temporary-locals
```

Figure 4.2 ZePoP: Phase1- Calculating the shortest path delays

Methods

```
adjustSend (s, y) // adjust # of sends to neighbor y
Begin
1. if $\Psi_{sy} \leftarrow \boldsymbol{true}$ then
2.        $O_{xy} \leftarrow O_{xy} - 1$
3.        $\Psi_{sy} \leftarrow \boldsymbol{false}$
End
```

```
Accept (E, s, y) // adjust # of receives from neighbour y
Begin
1. $G_s \leftarrow [G_s, y]$ // direction or gateway
2. if ($\pitchfork_{sy} = \boldsymbol{false}$) then // receipt flag of s via y is false
3.        $\pitchfork_{sy} \leftarrow true$
4.        $I_{xy} \leftarrow I_{xy} + 1$
5. $D_{sy} = E.D$
End
```

```
Forward (E, s, y) // better messages are forwarded
Begin
1. $E.D \leftarrow D_{sx}$ // update the message
2. for $z \in NB_x, z \neq y$
3.        If $D_{sz} > (D_{sx} + d_{xz})$ and $\Psi_{sz} = \boldsymbol{false}$ then
4.                $\Psi_{sz} \leftarrow \boldsymbol{true}$ // send flag
5.                $O_{xz} \leftarrow O_{xz} + 1$
6.        If $\Psi_{sz} \leftarrow \boldsymbol{true}$ then // forwarded flag true
7.                $D_{sz} \leftarrow D_{sx} + d_{xz}$ // update s to z path delay
8.        $E.d \leftarrow d_{xz}$
9.        Send (E, z)
End
```

```
ResetDirection(s) // the shortest path direct of node s
Begin
1. for $y \in G_s$
2.        $I_{xy} \leftarrow I_{xy} - 1$
3.        $\pitchfork_{sy} \leftarrow \boldsymbol{false}$ // reset receive the flag of s via y
End
```

```
ReceiveInform ($C'_x$, Leader, Leaderdirection) Begin
1. Receive (I, g) // receive INFORM message
2.        if I.C > $C'_x$ or I.C = $C'_x$ and Leader>I.nodeID then
3.                Leader $\leftarrow$ I.nodeID
4.                $C'_x$ $\leftarrow$ I.C
5.                Leaderdirection $\leftarrow$ g
End
```

Figure 4.3 Supporting methods of phase1

Phase1 overview: In the first phase of the algorithm, each node x broadcasts the ELECTION(E) message and calculates path delays D_{sx} upon receipt of ELECTION messages from all others node $s \epsilon V, s \neq x$. Each node continues processing ELECTION messages until it receives at least one message from all other nodes. Figure 4.2 shows the phase1 of the election algorithm, and Figure 4.3 shows the supporting methods. Table 4.1 describes the notations used in the algorithm with their initial values that are assigned before each election. When an ELECTION message of source *s* travels from the node *y* to its neighbor *x*, it carries the cumulative link delays or the path delay D_{sy} in the field D, as well as the link delay d_{yx} in the field 'd'. Upon receipt of that message, the node *x* calculates D'_{sx}(line3) and use it with $\phi_s$to decide whether to drop the message or process further. If the received message is the first copy from s, it sets ϕ_s, increment message count, calculates D_{sx}, $⫛_{sy}$ to **true**, and increment I_{xy} by 1. It also forwards the message to other neighbors z and increments $O_{xz}\ by$ 1 and sets Ψ_{sz} to **true** (line 4-9). While processing the next copies of ELECTION message from *s*, each time the node *x* updates I_{yx} and O_{xy} considering the values D_{sx}, D'_{sx}, D'_{sy}, $⫛_{sy}$ and Ψ_{sy} (line 10-23). I_{xy} and O_{xy} are considered as the **branch weight information**. During the updates, x classifies the node *s* in one of the three categories, as discussed below. The node *x* also identifies the slow direct links with the neighbors so that it can avoid them (marked as **dead links**) during communication (line 25-27). On the dead links, both I_{xy} and O_{xy} would be 0.

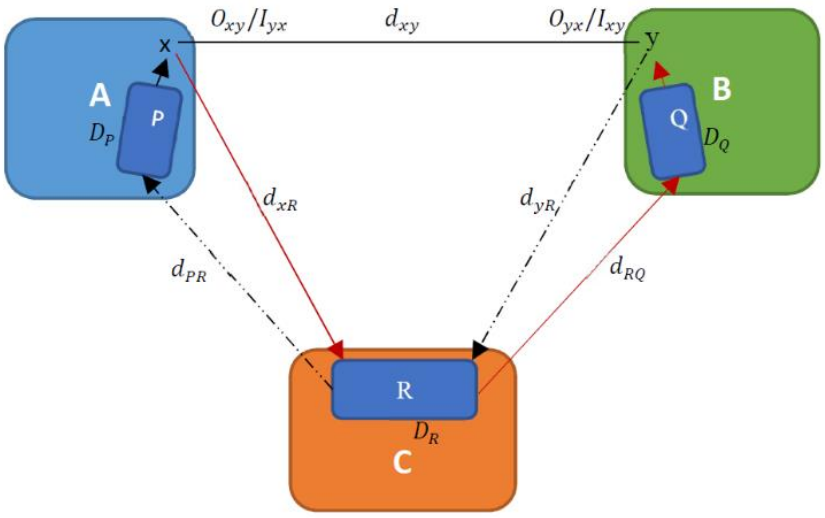


Figure 4.4 Node classification based on ELECTION messages

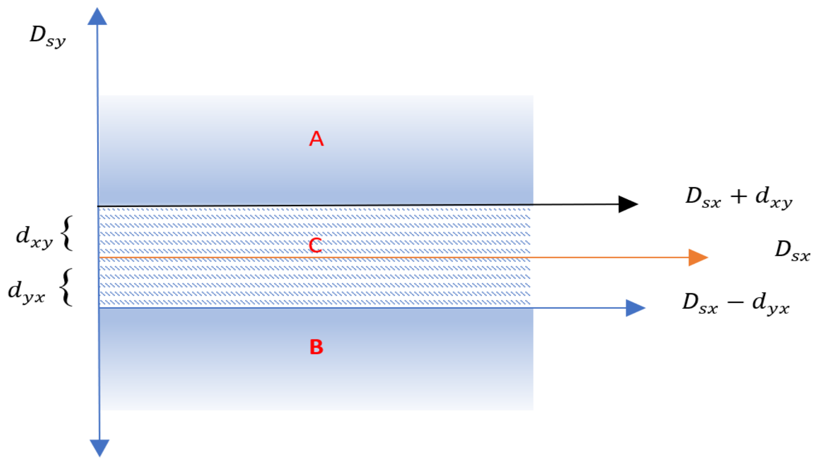


Figure 4.5 Node classification with respect to link (x, y)

Node Classification for Neighborhood Comparison:

As the node x receive election messages and updates I_{xy} and O_{xy} considering the values D_{sx}, D'_{sx}, D'_{sy}, $⫛_{sy}$ and Ψ_{sy}, it classifies each source node s with respect to the link (x,y) in one of the classes A, B, or C, as shown in Figure 4.4. The Node-Set A contains all participants, including x, such that they have the shortest path to y only via x. Similarly, nodes in B, including y, have the shortest-path to x only via y. Participant nodes in C have the shortest path to x or y without going through y or x, respectively. P, Q, and R are the subset of nodes from A, B, and C, respectively, that the paths between x and y use. If the network topology has no alternate path between x and y except the direct one, the classification is straight forward with C empty. All nodes reaching x via y are in set B, and the rest of the nodes are behind x, so they are in set A. If there are multiple alternative paths between x and y then some nodes $R \subseteq C$ will be along the paths. The route from x to y might also include some nodes $Q \subseteq B$ and y to x might include some nodes $P \subseteq B$. As a node x receives ELECTION message from other nodes via these different paths, it ***classifies s with respect to link (x, y)***. Finally, the x considers the node *s* in,

i. Set A, if $D_{sy} > D_{sx} + d_{xy}$, $D_{pq} is$ initialized ∞ . $D_{sy} = \infty$, if x has no copies of ELECTION message from s via y.

ii. Set B, if $D_{sx} > D_{sy} + d_{yx}$ or $D_{sy} < D_{sx} - d_{yx}$,

iii. Set C if $D_{sx} < D_{sy} + d_{yx}$ but $D_{sy} < D_{sx} + d_{xy}$ i.e. $(D_{sx} + d_{xy}) \leq D_{sy} \leq (D_{sx} + d_{xy})$

This classification is shown diagrammatically in Figure 4.5. Thus, for each live link (x, y) the node x knows that $O_{xy} = |A|$ and $I_{xy} = |B|$. Now, suppose T_x^C and T_y^C are the total delay from the nodes in C, to node x and y, respectively. They are defined as follows, $T_x^C = \sum_{c \epsilon C} D_{cx}$ and $T_y^C = \sum_{c \epsilon C} D_{cy}$. The algorithm aims to enable each node x to decide if it is a leader candidate based on neighborhood comparison of recorded values. For that, the node x also record both T_x^C and T_y^C as shown in lines 21 and 22. The algorithm uses the values of O_{xy}, I_{xy}, T_x^C and T_y^C to determine the leader candidates in phase2.

4.4.3 The election algorithm: Phase2

In the second phase, the algorithm first aims to reduce the number of leader candidates by using the recorded values in phase1. Then, it elects one of the few candidates as the new leader. Each node *x* checks if it is a better candidate for the leader compared to its neighbor y. For *x* to be a better candidate, it must satisfy the following condition,

$$\text{Closeness Centrality, } C_x > C_y$$

$$\text{Or } \frac{|V|}{T_x} > \frac{|V|}{T_y}$$

$$\text{Or } \boldsymbol{T_x < T_y} \tag{4.2}$$

Where the T_x and T_y are the total delay from all other nodes at x and its neighbor y, respectively. We can calculate them as,

$$\boldsymbol{T_x = T_A + T_B + T_x^C + I_{xy} d_{yx}} \tag{4.3}$$

$$\boldsymbol{T_y = T_A + T_B + T_y^C + O_{xy} d_{xy}} \tag{4.4}$$

Where the T_A and T_B are the total delays from the nodes in set A and B, to the node x and y, respectively. So, assuming $d_{xy} = d_{yx}$ and $d_{xy} > 0$, x must satisfy,

$$\boldsymbol{O_{xy} > \underbrace{I_{xy} + \frac{T_x^C - T_y^C}{d_{xy}}}_{I'_{xy}}} \tag{4.5}$$

```
Phase2: Leader election(x)
1.  candidacy ← false
2.  first_message ← false // everyone must know about a candidate
3.  if $\phi_{xy} = true, \forall_{y \in NB_x}$ then
4.          Candidacy ← true
5.  if candidacy = true then
6.        Calculate the closeness centrality $C_x$ as eq. (1)
7.        Broadcast INFORM containing $C_x$
8.         first_message ← true

9.  Leader ←x
10. $C'_x$ ← 0
11. t ← 0,
12. T ←k $* \max(D_{sx}, \forall_{s \in V})$// k is a constant
13. LeaderDirection ← x
14.  while first_message = false
15.        first_message= Check_Inform_arrival()// non-blocking check
16.        If first_message=true
17.             ReceiveInform($C'_x$, Leader, Leaderdirection)
18. while t<T // runs for diameter
19.       flag =false
20.       flag= Check_Inform_arrival()
21.      If first_message=true
22.             ReceiveInform($C'_x$, Leader, Leaderdirection )
23. FG_LIST ←{$NB_x - Leaderdirection$}
24. Send SUBS message to Leaderdirection
25. Send USUBS message to all in FG_LIST
26. CHILD_LIST ←{}
27.  for c=1 to $|NB_x|$
28.       Receive (US, k)
29.       If US. Type=SUBS then
30.                CHILD_LIST.add(k)
end
```

Figure 4.6 ZePoP: Phase2-Leader selection

$$\text{So, } \boldsymbol{O_{xy} > I'_{xy}} \tag{4.6}$$

However, there is a possibility that the d_{xy} would be zero. Then the condition becomes,

$$\boldsymbol{T_x^C < T_y^C} \tag{4.7}$$

Now, let's define a term δ_{xy} for node *x* as follows,

$$\boldsymbol{\delta_{xy}} = \begin{cases} \boldsymbol{O_{xy} - I'_{xy}}, & \boldsymbol{if\ d_{xy} > 0} \\ \boldsymbol{T_y^C - T_x^C}, & \boldsymbol{if\ d_{xy} = 0} \end{cases} \tag{4.8}$$

If the *superiority* of node x is ϕ_{xy} with respect to the neighbor y, then

$$\boldsymbol{\phi_{xy}} = \begin{cases} \boldsymbol{true\ \ if\ \delta_{xy} > 0} \\ \boldsymbol{false\ if\ \delta_{xy} < 0} \\ \boldsymbol{false\ if\ \delta_{xy} = 0\ and\ (y < x\ \&x! = L)\ or\ y = L} \end{cases} \tag{4.9}$$

Where L is the ID of the current leader. When $\delta_{xy} = 0$, both x and y are the equally better candidate, but the lowest ID or the existing leader breaks the tie.

Now, a node *x* can declare itself as a leader candidate only if $\phi_{xy} = true$ for all neighbors $y\epsilon\ NB_x$, (lines 3-4). For the election, all the candidates declare their closeness centrality to others through the INFORM message. Each node selects the node as the leader, whose INFORM message contains the highest centrality. Figure 4.6 shows the phase2 of the algorithm. At the end of the phase2, all nodes exchange SUBS or USUBS string among the direct neighbors to form the DCDT. Each node knows its current parent and the children in *CHILD_LIST*. The root of the DCDT is the optimally placed leader. Thus, the DCDT is a delay-balanced tree and ready for any delay-sensitive data communication. A node can avoid getting disconnected from DCDT by picking another node from *FG_LIST* as the parent in case the current parent leaves.

4.4.4 Explanation by Example

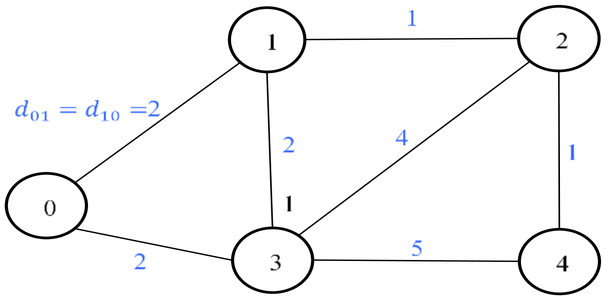


Figure 4.7 An Example topology

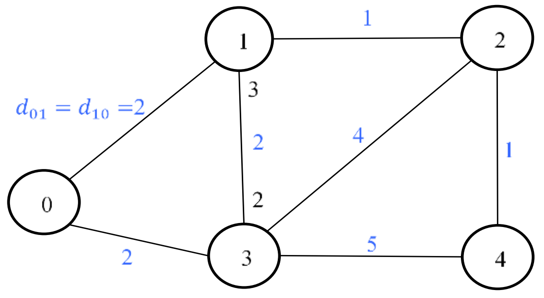
Table 4.2 Initial states in node 1 and 3

	Node 1						Node 3				
G	-	1	-	-	-	G	-	-	-	3	-
	0	1	2	3	4		0	1	2	3	4
D	∞	0	∞	∞	∞	D	∞	∞	∞	0	∞
	0	1	2	3	4		0	1	2	3	4
ϕ	false	true	false	false	false	ϕ	false	true	false	true	false
	0	1	2	3	4		0	1	2	3	4
$T_1^C = 0; T_3^C = 0$						$T_1^C = 0; T_3^C = 0$					

Let us consider the example topology shown in Figure **4.7**. The numbers on the edges are the link delays. In the first phase, when ELECTION messages are broadcasted, any two neighbors (x, y) can calculate O_{xy}, I_{xy}, T_x^C *and* T_y^C based on node classification. Let us discuss the node classification for node pair or link (1, 3). So, We are interested in calculating O_{13}, I_{13}, T_1^C *and* T_3^C at 1 and O_{31}, I_{31}, T_3^C *and* T_1^C at 3.

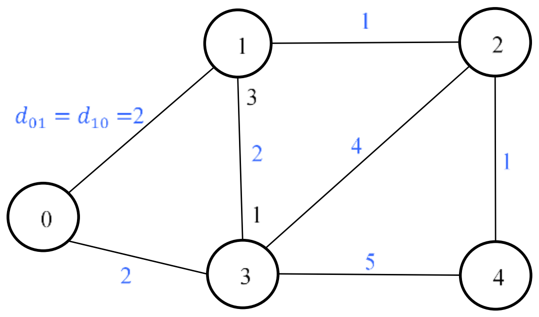
Table 4.2 shows the initial values of local variables at both nodes 1 and 3. Now the calculation of the desired variable based on node classification is shown in multiple rounds of ELECTION message processing. The ELECTION message is shown as ***E (source, direction, path-delay).***

Round1: *Receive messages from some direct fast links*

Message Processing	**Updating diagram and local variables after processing**
Node 1: receives E (0, 0, 2), E (2, 2, 1), E (3, 3, 2) As ϕ_0, ϕ_2=false, it forwards E (0, 0, 2), E (2, 2, 1) towards node 3 and updates the local variables (Line 4-9) Sets: $A_1 = \{0, 1,2\}, B_1 = \{3\}, C_1 = \{\}$ Values: $O_{13} = \lvert A_1 \rvert = 3$ $I_{13} = \lvert B_1 \rvert = 1$ $T_1^C = 0$ $T_3^C = 0$	 1 2 1 3 $d_{01} = d_{10} = 2$ 2 4 1 0 2 2 5 3 4 2 Updates at 1: G $\begin{array}{\|c\|c\|c\|c\|c\|} \hline 0 & 1 & 2 & 3 & - \\ \hline \end{array}$ $\quad\ \ \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array}$ D $\begin{array}{\|c\|c\|c\|c\|c\|} \hline 2 & 0 & 1 & 2 & \infty \\ \hline \end{array}$ $\quad\ \ \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array}$ ϕ $\begin{array}{\|c\|c\|c\|c\|c\|} \hline \text{true} & \text{true} & \text{true} & \text{true} & \text{false} \\ \hline \end{array}$

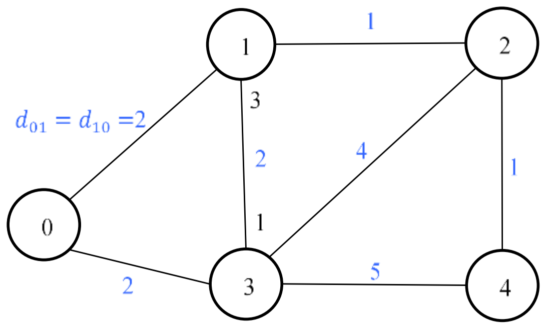
Node 3: receives E (0, 0, 2), E (1, 1, 2) As ϕ_0=false, it forwards E (0, 0, 2) towards node 1 and updates the local variables (Line 4-9) Sets: $A_3 = \{0,3\}, B_3 = \{1\}, C_3 = \{\}$ Values: $O_{31} = \|A_3\| = 2 \;\; I_{13} = \|B_1\| = 1$ $T_1^C = 0$ $T_3^C = 0$	$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array}$ Updates at 3: $\begin{array}{c\|c\|c\|c\|c\|c\|} G & 0 & 1 & - & 3 & - \\ & 0 & 1 & 2 & 3 & 4 \end{array}$ $\begin{array}{c\|c\|c\|c\|c\|c\|} D & 2 & 2 & \infty & 0 & \infty \\ & 0 & 1 & 2 & 3 & 4 \end{array}$ $\begin{array}{c\|c\|c\|c\|c\|c\|} \phi & \text{true} & \text{true} & \text{false} & \text{true} & \text{false} \\ & 0 & 1 & 2 & 3 & 4 \end{array}$

Round 2: *Receive some multi-hop messages via the fast links*

Message Processing	**Updating diagram and local variables after processing**
Node 1: receives E (0, 3, 4), E (3, 0, 4), E (4, 2, 2) E (0, 3, 4): It is discarded but as it learns that node 3 is in an equally better position for node 0, It moves node 0 to set C. (line 19-23) E (3, 0, 4): It is simply discarded as a better message has already been received via direct link. E (4, 2, 2): As ϕ_4=false, it is forwarded towards node 3, and local variables are updated (Line 4-9) Sets: $A_1 = \{1, 2. 4\}, B_1 = \{3\}, C_1 = \{0\}$ Values: $O_{13} = \|A_1\| = 3 \;\; I_{13} = \|B_1\| = 1$ $T_1^C = 2$ $T_3^C = 2$	 1 2 0 3 4 1 3 $d_{01} = d_{10}$ =2 2 4 1 1 2 5 Updates at 1: $\begin{array}{c\|c\|c\|c\|c\|c\|} G & 0 & 1 & 2 & 3 & 2 \\ & 0 & 1 & 2 & 3 & 4 \end{array}$ $\begin{array}{c\|c\|c\|c\|c\|c\|} D & 2 & 0 & 1 & 2 & 2 \\ & 0 & 1 & 2 & 3 & 4 \end{array}$

Node 3: receives E (0, 1, 4), E (2, 1, 3) E (0, 1, 4): is discarded but move the node 0 to class (line 19-23) E (2, 1, 3): As ϕ_2=false, it forwards E (2, 1, 3) towards other neighbors (except 1) and updates local variables (Line 4-9) Sets: $A_3 = \{3\}, B_3 = \{1{,}2\}, C_3 = \{0\}$ Values: $O_{31} = \lvert A_3 \rvert = 1$ $I_{13} = \lvert B_1 \rvert = 2$ $T_1^C = 2$ $T_3^C = 2$	$\begin{array}{c\|c\|c\|c\|c\|c\|} \phi & \text{true} & \text{true} & \text{true} & \text{true} & \text{true} \\ & 0 & 1 & 2 & 3 & 4 \end{array}$ Updates at 3: $\begin{array}{c\|c\|c\|c\|c\|c\|} G & 0 & 1 & 1 & 3 & - \\ & 0 & 1 & 2 & 3 & 4 \end{array}$ $\begin{array}{c\|c\|c\|c\|c\|c\|} D & 2 & 2 & 3 & 0 & \infty \\ & 0 & 1 & 2 & 3 & 4 \end{array}$ $\begin{array}{c\|c\|c\|c\|c\|c\|} \phi & \text{true} & \text{true} & \text{true} & \text{true} & \text{false} \\ & 0 & 1 & 2 & 3 & 4 \end{array}$

Round3: *Receive some more messages, including the message from the slow links.*

Message Processing	**Updating diagram and local variables after processing**
Node 1: Exits phase 1 with Sets: $A_1 = \{1, 2. 4\}, B_1 = \{3\}, C = \{0\}$ Values: $O_{13} = \lvert A_1 \rvert = 3$ $I_{13} = \lvert B_1 \rvert = 1$ $T_1^C = 2$ $T_3^C = 2$	 1 2 0 3 4 $d_{01} = d_{10} = 2$ 1 3 2 1 4 1 2 5 Updates at 1: $\begin{array}{c\|c\|c\|c\|c\|c\|} G & 0 & 1 & 2 & 3 & 2 \\ & 0 & 1 & 2 & 3 & 4 \end{array}$

D	2	0	1	2	2
	0	1	2	3	4

ϕ	true	true	true	true	true
	0	1	2	3	4

Node 3:
receives E (2, 2, 4), E (4, 1, 4), E (4, 4, 5)
E (2, 2, 4): It has come from node 2, but already the fastest message from 2 has been received via 1. So, discarded.
E (4, 1, 4): As ϕ_4=false, it forwards E (4, 1, 4) towards other neighbors (except 1) and updates local variables. (Line 4-9)
E (4, 4, 5): discarded

Sets: $A_3 = \{3\}, B_3 = \{1,2,4\}, C_3 = \{0\}$
Values: $O_{31} = |A_3| = 1$ $I_{13} = |B_1| = 3$
$T_1^C = 2$
$T_3^C = 2$

Updates at 3:

G	0	1	1	3	1
	0	1	2	3	4

D	2	2	3	0	4
	0	1	2	3	4

ϕ	true	true	true	true	true
	0	1	2	3	4

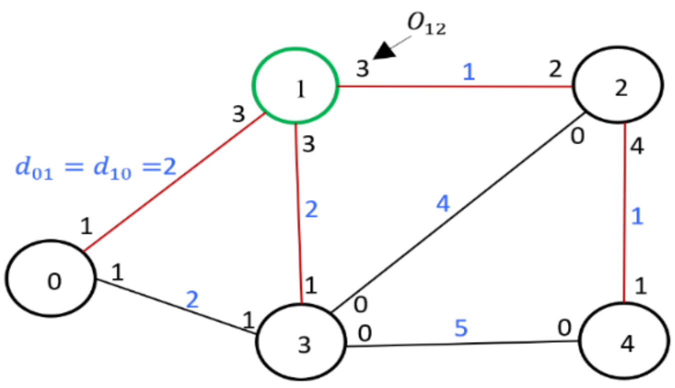


Figure 4.8 The example topology with the final numbers. The red edges form the DCDT rooted at leader 1

Thus, for any link (x, y) O_{xy}, I_{xy}, T_x^C and T_y^C can be calculated. The final diagram with these numbers for all links are shown in Figure **4.8**. For example, $O_{1,2} = 3$ means 3 ELECTION messages (for nodes set {0, 1, 3}) to node 2 will find the shortest path via node 1. In phase2, node 1 will find $\phi_{1,3} = \boldsymbol{true}$ but node 3 will find $\phi_{3,1} = \boldsymbol{false}$ according to the equations from (4.5) to 4.9). Thus, only node 1 will see $\phi_{1,j} = \boldsymbol{true}$ for all its neighbors j. Therefore, node 1 is the only candidate for leader, and eventually, others elect it. According to the equation (4.1), the closeness centrality at the leader candidate 1 is **0.714**. The closeness centrality at nodes 0, 2, 3, and 4 are 0.45, 0.62, 0.45 and 0.45 respectively. The red edges form the DCDT rooted in the optimally placed leader node 1. The links (2,3) and (3, 4) are **dead links** because $O_{2,3} = O_{3,2} = 0$, $O_{3,4} = O_{4,3} = 0$.

4.4.5 Minimum-Cost Spanning Tree(MST) vs. DCDT

For leader election in the weighted network, where weight is the speed of the communication link, a very commonly used solution is: create an MST and then pick any node or the center of the tree as the leader. However, the MST cannot guarantee the highest closeness centrality. Let's consider the example in Figure **4.8**. The tree shown using the red lines is the DCDT, as well as an MST. However, if node 3 picks the link (3,0) instead of (3,1), the tree is still an MST but not the optimal DCDT. In that case, node 1 is still the center of the MST, but the closeness centrality would be 0.55, which is much lower than closeness centrality 0.714 of DCDT.

4.4.6 Supporting Algorithms in the Protocol.

As the protocol is for the P2P dynamic network, it must handle the new node arrival as well as node departure. Upon node arrival or departure, the network topology changes. So, the leader must adjust the location based on the topology changes. In CHAPTER 3, we have already discussed the algorithms for node joining, node departure, and election initiation. The first two algorithms can be directly used for this protocol. For election initiation, we found MCV is better than the CBW. So, the MCV can be used in this protocol by modifying test criteria, i.e., at leader i, check for violation of leader candidacy condition and it should start the election for location adjustment if for any neighbor j, $\phi_{i,j}$ becomes ***false***.

4.4.7 Guaranteed Delivery

In distributed systems, the terms "at-least-once" "at-most-once" and "exactly-once" frequently come when the guaranteed message delivery is discussed. The ZePoP needs "at-least-once" delivery to all because otherwise, we might have cases where there is no leader candidate, i.e. each node x will fail to satisfy $O_{xy} \geq I_{yx}, \forall_{y \in NB_x}$. Figure 4.9 shows such a case where no one can be the leader candidate because of multiple message drops. So, for the guaranteed delivery of ELECTION messages, the control layer must be implemented on TCP. Since each node waits to receive at least one message from others, we assume that no node leaves during the election process.

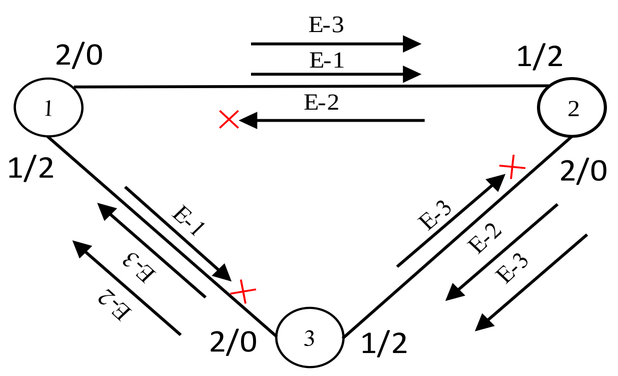


Figure 4.9 No leader candidate situation because of message drops**.**

Theorem 4.1 ***The election protocol always finds a leader node maximizing the delay-based closeness centrality.***

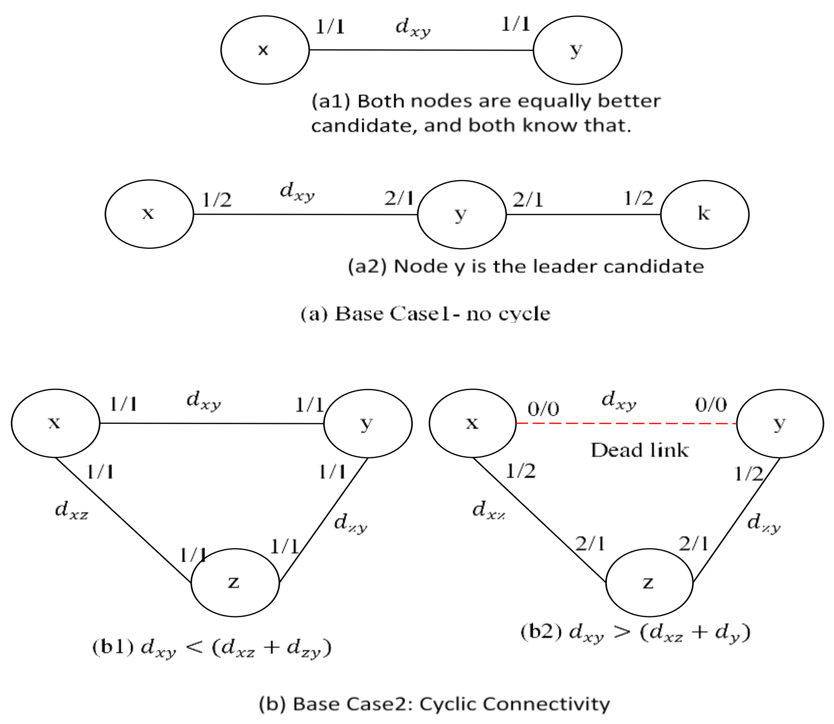

x
1/1
d_{xy}
1/1
y
(a1) Both nodes are equally better candidate, and both know that.
x
1/2
d_{xy}
2/1
y
2/1
1/2
k
(a2) Node y is the leader candidate
(a) Base Case1- no cycle
x
1/1
d_{xy}
1/1
y
1/1
1/1
d_{xz}
d_{zy}
1/1
1/1
z
(b1) $d_{xy} < (d_{xz} + d_{zy})$
x
0/0
d_{xy}
0/0
y
Dead link
1/2
1/2
d_{xz}
d_{zy}
2/1
2/1
z
(b2) $d_{xy} > (d_{xz} + d_y)$
(b) Base Case2: Cyclic Connectivity

Figure 4.10 Topology Base Cases and leader candidacy

Proof: First, we derive all possible network scenarios from Figure 4.4 and then show that each scenario has at least one leader candidate.

Base Case1- No cycle: class C is empty. There is only one node x in A and one node y in class B. So, only two nodes in the system, Figure 4.10 (a1). Both nodes are equally better candidates for leader, the lowest ID breaks the tie. But if we have a neighbor of y in B, then y is a better candidate than x, for any positive value of d_{yk} Figure 4.10 (a2).

Base Case2- Cyclic connectivity: A single node from each class. Because of random delays on the links, we can have few possible subcases. For every pair of nodes x, y, (i) if $d_{xy} < (d_{xz} + d_{zy})$, where z is the third node in the cycle, then all nodes are equally better candidates, if $d_{xy} = d_{xz} = d_{zy}$ because $\phi_{i,j} = true$ for each *i,j* pair, Figure 4.10 (b1). But if $d_{zy} > d_{xz}$ and $d_{xz} \geq d_{xy}$ then only x is the leader candidate as both $\phi_{x,y} = true$ and $\phi_{x,z} = true$. (ii) But if $d_{xy} > (d_{xz} + d_{zy})$, then the direct link between x and y is considered as a **dead link,** Figure 4.10 (b2). On dead link (x, y), $O_{xy} = I_{xx} = 0$. In this case as well as when $d_{xy} = (d_{xz} + d_{zy})$, node z will be the leader candidate,

General case1- No cycle: The topology is a tree structure. So, class C is empty for any link (x, y). We have one or more other nodes in A and B. In the tree structure, there is a single-center if it is a centered tree or two adjacent centers for bicentered tree. For single-center x, $\phi_{x,v} = true$ will be true for all neighbor v. So, x is the leader candidate. For the bicentered tree, two centers x and y are neighbors and $O_{xy} = I_{xy}$ is true. So, both x and y can be the candidates, but the algorithm chooses the lower node ID.

General Case2 -Cyclic Connectivity: There are multiple paths between x and y, including the direct link, Figure 4.4. The alternate paths can take 0 to many nodes from P, Q but at least one from R. Then we can locate the leader candidates as follow:

- if $O_{xy} = I_{xy}$ and $T_x^C = T_y^C$ then both set A and B are in the equal position to contain the leader candidate only if for any node $z\epsilon C$, $\phi_{xz} = true$ and $\phi_{yz} = true$. The set C also would have a candidate if $\phi_{zp} = true$ as well as $\phi_{zq} = true$ for any $p\epsilon A$ and $q\epsilon B$. Figure 4.10-b2 is a special case for that.
- Now, if $\phi_{xy} = true$ and for any node $z\epsilon C$ $\phi_{xz} = true$ then the leader candidate is in set A. Similarly, it is possible to find any set B or C, where the leader candidate(s) would exist.

After identifying the set for candidacy, we can move within the set towards the link, say (x, q) where $\phi_{xq} = false$ until such a q exists. $\phi_{xq} = false$ means q is in a better position than x. This movement cannot be infinite if there is no message drop; i.e., we never complete a cycle. In other words, we can never have a general Case2 where y is better than x ($\phi_{xy} = false$), z is better than y ($\phi_{yz} = false$) and x is better than z ($\phi_{zx} = false$). Here $x\epsilon A, y\epsilon B\ and\ z\epsilon C$. Eventually, we must reach an equilibrium position like the cases mentioned above where we have at least one leader candidate, and the candidate node, say x, sees for each neighbor j, $\phi_{xj} = true$.

So, the algorithm always finds very few but at least one leader candidate. As each candidate x ensures $\phi_{xj} = true$ for all neighbors $j\epsilon\, NB_x$, they are best in terms of closeness centrality compared to their neighbors. When each candidate informs its closeness centrality to others via INFORM message, they always elect the candidate whose INFORM message contains the highest closeness centrality.

4.4.8 Leader Election Validation Criteria

The ZePoP meets all the different criteria of a valid leader election protocol, as discussed below:

1. *Termination:* Each node terminates the first phase after receiving *n-1* ELECTION messages and the second phase after receiving one or more INFORM messages.
2. *Uniqueness:* In Phase2, lines 14-22, each node, including the candidates, receives the closeness centrality of all candidates via the INFORM message. The candidates with low closeness centrality give up the candidacy, and eventually, only one candidate claims the leadership.
3. *Agreement:* Each node elects the candidate with the highest closeness centrality. In the case of equal centralities of multiple candidates, the protocol uses the node IDs to break the tie. (method *ReceiveInform*).

4.5 Experimental Results and Discussion

For experimental validation, the protocol is implemented as a distributed application using MPI and C++. For running the application, P2P overlay networks are generated randomly on the local cluster nodes. ***The link delays are estimated in the overlay by 3-way message communication.*** These delays are directly used in the leader election algorithm. The protocol is run for different random overlay networks. Figure 4.11 shows one of these networks. It has 50 nodes, i.e., n=50. The number pairs on the links are O_{xy} - O_{yx} , recorded by the protocol, where $x < y$. For example, between the node 3 and 7, the pair is 6-43. So, $O_{37} = 6$ and $O_{73} = 43$. In words, the node 3 is nearest to 6 nodes, where node 7 is nearest to 43 nodes. In this scenario, node 16 is the only leader candidate satisfying $\phi_{16y} = true$ for all neighbors y in {13, 14, 19}. So, 16 is elected as the leader by all other nodes. The DCDT is generated by the protocol rooted at the leader, and the red edges show it.

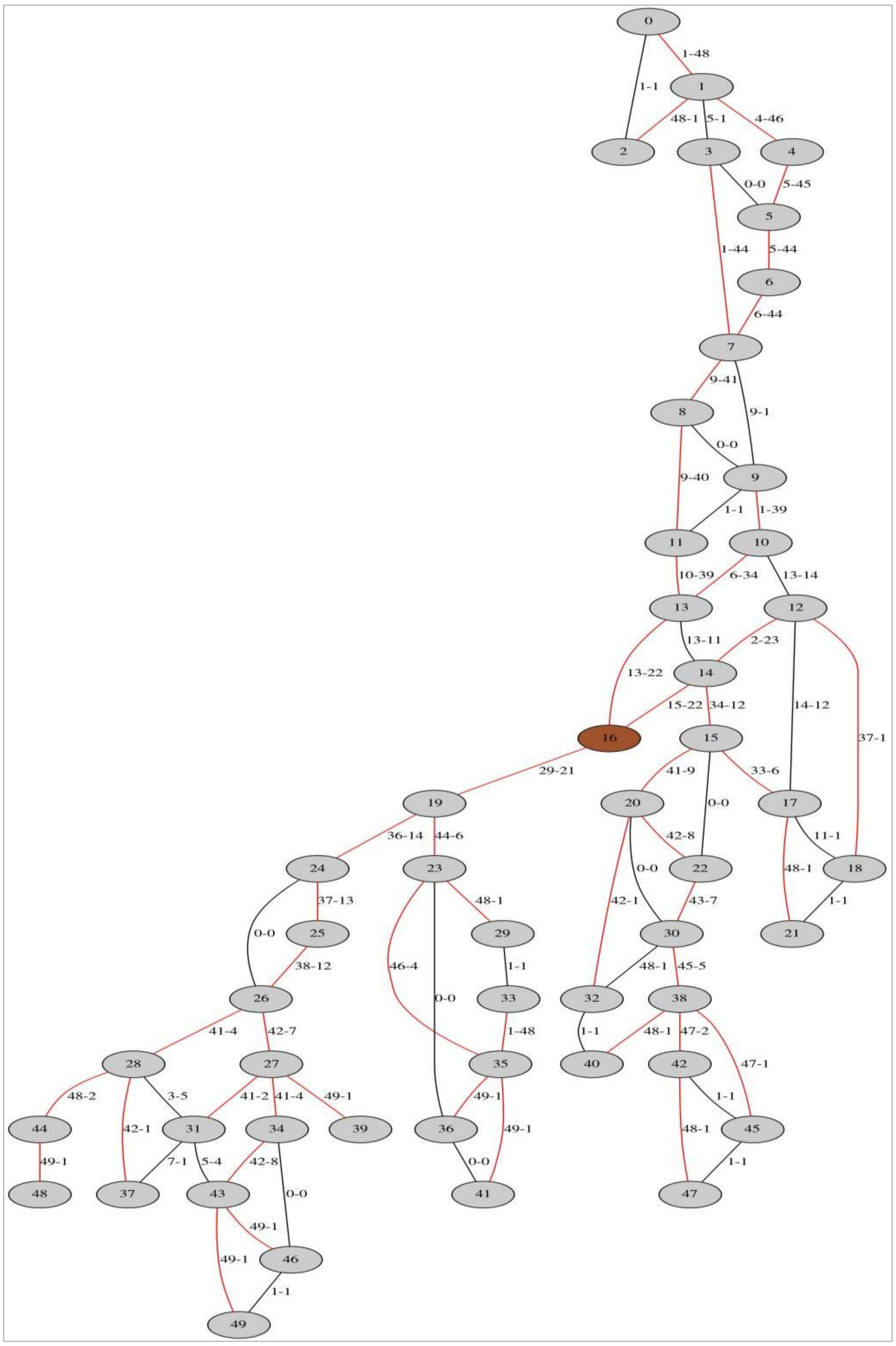


Figure 4.11 An overlay network generated to emulate the ZePoP protocol

4.5.1 Comparison of Closeness Centrality

For comparison, the closeness centrality of each node of the network is recorded after the election. Figure 4.12 shows these closeness centralities as a stem chart. The closeness centrality at the elected leader 16 is greater or equal to the centrality at all other nodes. Thus, the ZePoP protocol minimizes the average or mean delay from all the nodes to the DCDT root (leader). Therefore, it can maximize the DRDI at the leader.

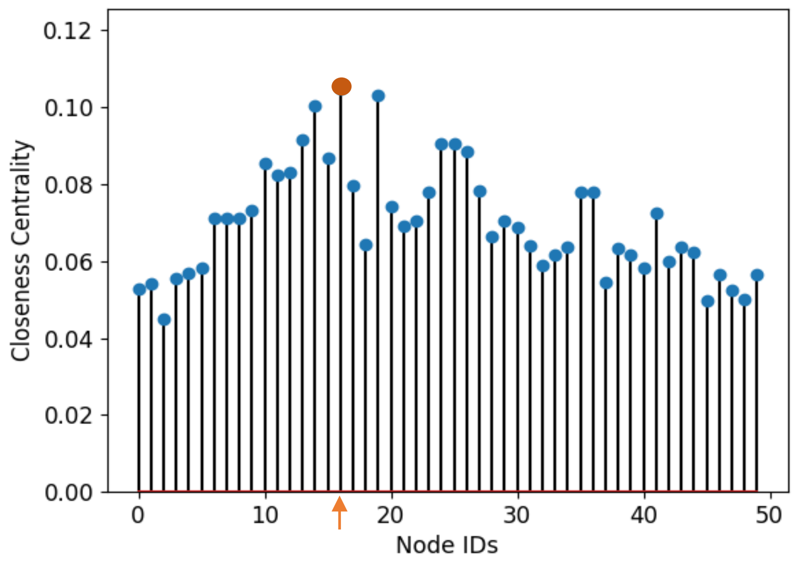


Figure 4.12 Closeness Centrality comparison among all nodes

4.5.2 Comparison of Vertex Eccentricity

The eccentricity of a vertex or a node is defined as the maximum distance from that node to all other nodes in the graph. In this work, the maximum distance would be in terms of delay. For comparison, the eccentricity (maximum delay) of each node is recorded during the election using ZePoP.

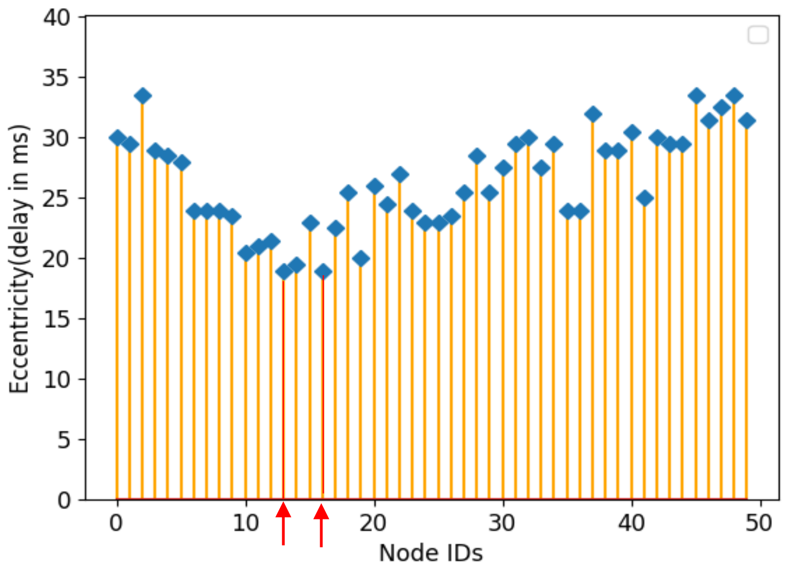


Figure 4.13 Comparison of eccentricity among all nodes

As we can see in Figure **4.13**, the eccentricity is minimum at node 16 as well as 13. However, the eccentricity of the leader will be near the minimum but not guaranteed to be the minimum.

4.5.3 Closeness Centrality vs. Eccentricity

The closeness centrality and eccentricity can be used to estimate the timing constraints of different applications to be run using DCDT. At each round of data collection and distribution on DCDT, all nodes must deliver data satisfying the timing constraints, i.e., data must be delivered within the pre-decided waiting period at the leader. But which one to use for modelling the waiting time is application dependent. If the application needs data from all nodes in every round, then the eccentricity of the leader plus a fixed margin can be used as the waiting time. However, there might have very few slow links, and some nodes have no path without using these slow links to deliver the data to the leader. These slow links would have a significant impact on the eccentricity at each node, including the leader, and the application might suffer from the scalability problem.

If the application can tolerate some data loss from some nodes, then we can use the inverse of closeness centrality or equivalently the average delay to speed up the data collection and distribution process by not waiting for the nodes behind the slow links. Suppose the shortest path delays from the nodes to the leader follows the normal distribution with mean μ and standard deviation σ. Then we can simply set a waiting time

$$W = \boldsymbol{\mu + 3\sigma} \tag{4.10}$$

The waiting time can ignore slow links or nodes and guarantee to receive data of almost 99% of the nodes. We deduce some more useful schemas of waiting time management later, but we keep the discussion limited here. Several wait-time management heuristics can be found in [81].

4.5.4 Application: Telepresence System

The ZePoP can elect an optimally positioned leader for the network topologies with variable link delays. If the selected leader is node M, then the protocol makes sure that the closeness centrality C_M of node M is maximum. In CHAPTER 3 we have seen that for maximizing DRDI or the profit function at the MC, we need to minimize the mean delay μ_M from all other nodes to the MC node. From equation 3.6) and (4.1) we can write,

$$\boldsymbol{\mu_M} = \frac{\mathbf{1}}{\boldsymbol{C_M}} \tag{4.11}$$

So, maximizing the closeness centrality, the protocol ZePoP can minimize the μ_M. Therefore, the leader elected by the protocol can be used to elect the MC node of P2P-based CMTS, maximizing its DRDI.

To show the application of the protocol, the leader elected by the ZePoP is considered as the MC of the P2P Telepresence System. So, it works as both MC as well as MP with assumptions 2, 3 and 4. Note that the DCDT created by the ZePoP is a generalized version of SCDT. The P2P telepresence system or CMTS is implemented as a distributed application using MPI and C++. The system is emulated on a local cluster with n = 50. The application starts with the node having ID 0 (assigned from MPI) and nodes *1, 2, ⋯, n-1* sequentially join the conference after every *15* seconds. The application is forced to create the same final topology, as shown in Figure 4.11. As each new participant comes, the current leader or MC initiates the re-election to position the MC optimally and maintain the balanced DCDT. We could also use the election initiation mechanism MCV for determining the new election requirement.

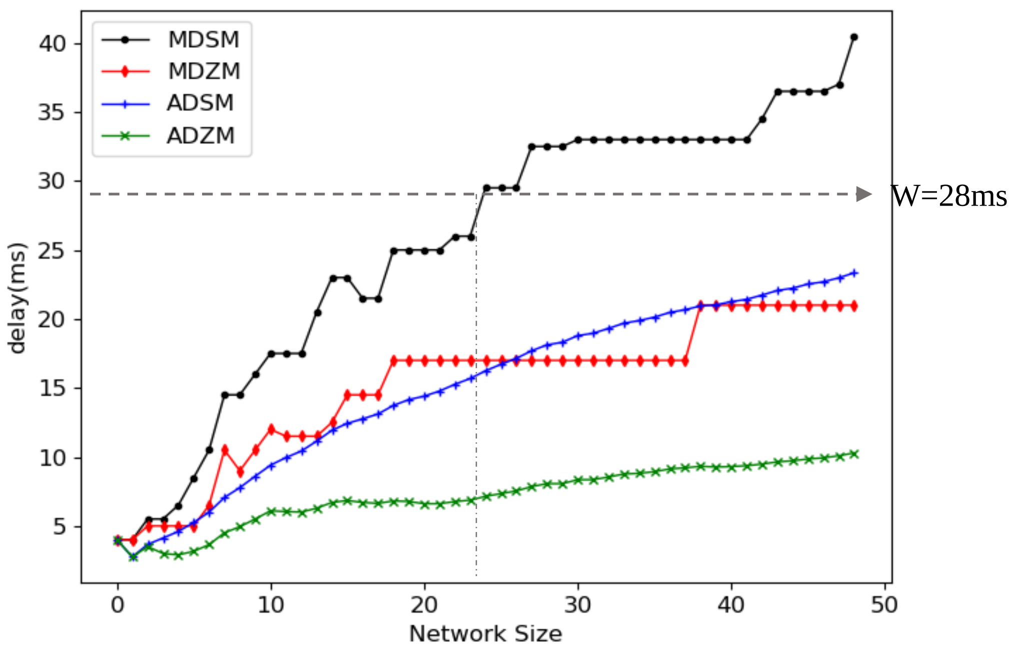


Figure 4.14 Comparison of delays between the ZePoP MC and the static MC

For the experiment, dummy video frames of size 1KB with header 23 bytes' header are transmitted. Each node sends its video streams towards the MC using the DCDT. The elected MC generates a composite dummy video of the same size by taking a portion of everyone's video packet and returns it to all nodes using the same DCDT. To show the benefit of centrality-based MC placement, we consider node 0 as the static MC. Then the average delays (inverse of closeness centrality) and maximum delays(eccentricity) are compared between the static MC and the ZePoP MC (dynamic MC elected by the ZePoP protocol). Figure 4.14 shows how these two kinds of delays increase as the network size grows. The four lines are for (i) Maximum Delays of Static MC(MDSM) (ii) Maximum Delays of ZePoP MC(MDZM) (iii) Average Delays at Static MC(ADSM) and (iv) Average

Delays at ZePoP MC(ADZM). The diagram shows that both average and maximum delays stay significantly low for the ZePoP MC compared to the Static MC case. When the network size is 50, the standard deviation of path delays from all nodes to the ZePoP MC was ~ 6.0. So, according to the equation (4.10), if the waiting time is set to ~10+3*6 = 28, then the ZePoP should be able to collect the streams of all participants before the wait-time expires, Figure 4.14. The same waiting time at the static MC will miss the streams of more than 50% of the participants. Thus, ZePoP MC can maximize DRDI, including more participants in the bounded time stream composition. However, during video communication, the links are shared by the multiple streams and the applications. The bitrate also can be different for the streams. So, for using the equation (4.10), the mean μ and σ must be updated in real-time for setting the waiting time. A couple of wait-time management schemes are discussed in later chapters of this dissertation.

ZePoP can also be compared with GAncestor in terms of total traffic, hotness of intermediate nodes, and the composition time in the context of the Telepresence System. For comparison, the telepresence or CMTS session is run using GAncestor, ZePoP and Static MC settings. For all three settings, the application is guided to form the same logical topology (Figure 4.11) incrementally as new nodes join. For each size of the network, total traffic, hotness of the intermediate nodes, and the composition time are recorded.

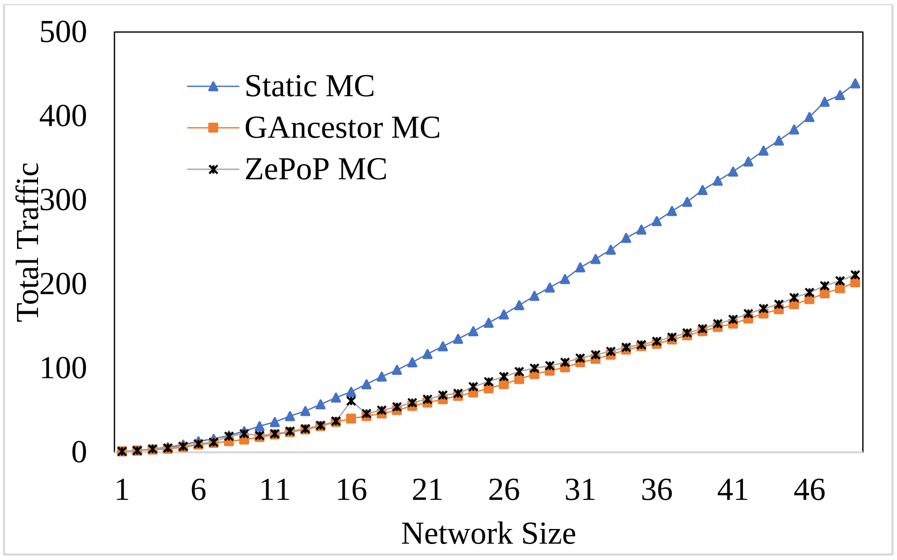


Figure 4.15 Comparison of total traffic

Figure 4.15 compares the total traffic between the three system settings for the telepresence system. Both the GAncestor and the ZePoP significantly reduce the total traffic, i.e., total hop count in the system. GAncestor is slightly better in terms of total traffic, which is expected. Because GAncestor aims to minimize hop-count, so it always picks direct links to travel from node x to y. But the ZePoP picks a multi-hop shortest path if the direct link is slow. The difference can be big in many situations if multiple direct links are very slow compared to their many hop alternate shortest paths. Because of the same reasons, the average and maximum path delays are expected to be higher for GAncestor MC compared to ZePoP MC.

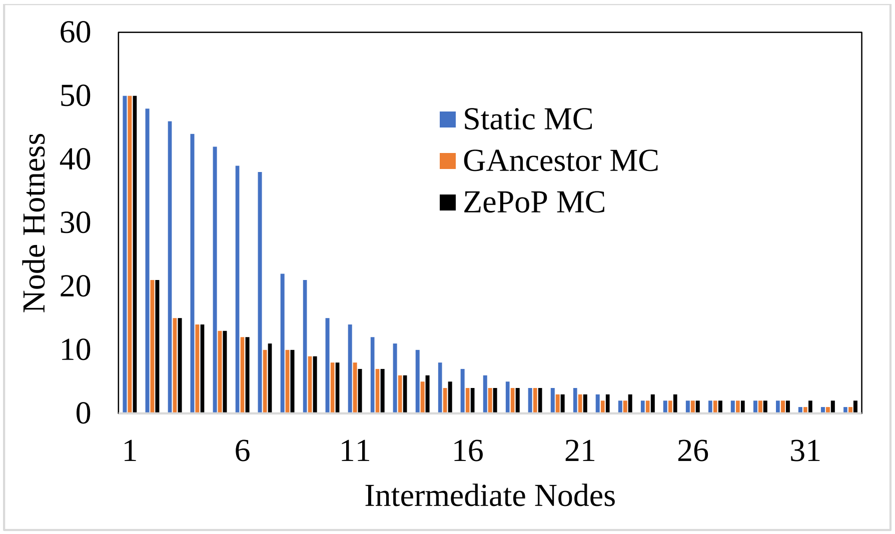


Figure 4.16 Comparison of hotness of intermediate nodes

Figure 4.16 shows the hotness of intermediate nodes in the SCDT for all three cases. As usual, the hotness or load is extremely high for intermediate nodes in the Static MC case. Because of the nature of our test topology, the order of hotness is almost the same for GAncestor and ZePoP MC cases. As the ZePoP selects alternate multi-hop paths for the streams, a stream will go through more nodes to reach the destination compared to the GAncestor. So, in ZePoP, more nodes will have high hotness compared to GAncestor MC case. This can be observed slightly in the second half of the chart.

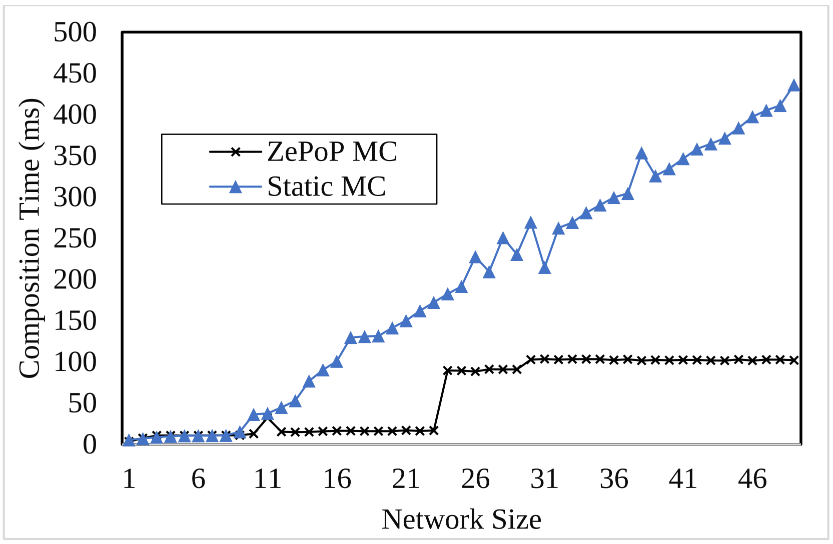


Figure 4.17 Comparison of composition time as the network grows.

In the last chapter, we have already seen that the composition time can be extremely high for Static MC compared to the GAncestor MC. From a similar experiment, the comparison based on the average composition time is shown in Figure 4.17 between Static MC and the ZePoP MC. In this setting, we found a similar result, i.e., the average composition time gets very high Static MC compared to the ZePoP MC as the network size grows. The composition time shown here is real and calculated from the emulation of CMTS. The composition time depends on many factors and parameters such as the current link delays, frame or stream rate, nodal delays, etc. If the network parameters change, the average composition time changes.

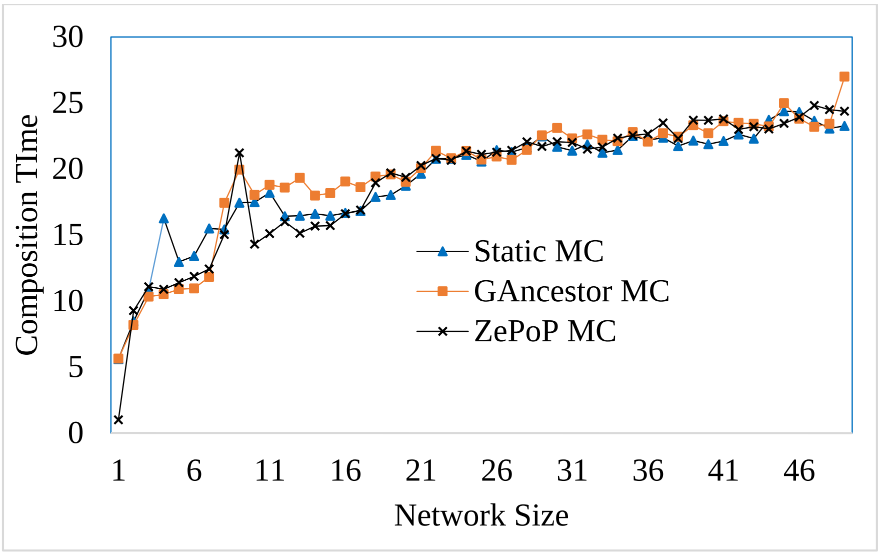


Figure 4.18 Comparison of the composition time

In another experiment setup, almost similar composition time values are achieved for all three schemes, Static MC, GAncestor MC, and ZePoP MC. For the topology under consideration, the ZePoP and GAncestor dynamically move the MC and the elected MC nodes for both cases are almost the same. Therefore, the composition time values are also nearly the same. However, for a different topology with lots of slow links, the difference in composition time can be very high because the ZePoP will use the fastest path. It was difficult for the Static MC setting to complete the simulation and generate some finite composition time values. Because node 0, which is considered as the Static MC, is sitting very far from most of the nodes. The node near the MC, for example, 1, 5 and 7, have very high hotness. The hotness of node 1 is 48, which is too high because it had to process continuous packets of size 3KB from 48 nodes. Many times, the application crashed with buffer overflow messages. After adjusting different parameters of the system, such as reducing bitrate, we achieved the composition time values as shown in the plot. However, every composition missed some participants, which can be referred to as the loss of profit. The concept of loss is discussed more in the next chapters. For ZePoP and the GAncestor, the hotness values for the node next to the MC were near 20, which is low compared to the static MC but still high for processing all high-bitrate streams. So, the load or hotness must be distributed among other nodes so that the composition time can be significantly reduced, i.e., the system profit is maximized. The load distribution of MC and nodes its vicinity is discussed in the next chapters

4.5.5 Message Complexity

In the first phase, each node broadcasts and always forwards the better ELECTION messages to the neighbors. However, we can use randomly filtered broadcasting to reduce some ELECTION messages in the network [82]. In the leader algorithm of ZePoP, each node requires only one copy of the ELECTION message from others. So, it is possible to further minimize the ELECTION messages in the network by forwarding only one copy of the ELECTION message for each node. However, the message complexity would still be O(nE). In the future, we plan to improve the message complexity by allowing ELECTION messages to travel up to a certain hop distance in the network.

4.5.6 Number of leader candidate

The leader election is considered as the symmetry breaking technique. So, the lowest number of leader candidates is better. For the test topologies, it is observed that the number of leader candidates in ZePoP is a maximum of 3, and that was only once among 50 elections. Among other cases, there were 2 candidates once as well, and in the rest of the elections, the number of leader candidates was only 1. It is also observed that as the network size grows, the number of leader candidates always stays 1. Thus, the protocol significantly reduces the number of messages in the second phase, as only one node will broadcast INFORM message.

4.6 Conclusion

In this chapter, ZePoP, a generalized leader election protocol, is discussed for the P2P dynamic network. The protocol picks one of the nodes dynamically that maximizes the delay-based closeness centrality. After each election, the algorithm creates an optimal multicast tree called DCDT rooted at the leader. The DCDT can be used in multiple applications for delay-constraint data collection and distribution. The protocol is the generalized version of GAncestor protocol discussed in CHAPTER 3. So, some algorithms are directly used in the protocol, especially for dynamic network management. The protocol is validated both theoretical proof as well as with experimental results. The experimental result shows that the ZePoP can improve the system scalability as well as application performance by optimizing the average path delays. The benefit of the leader election protocol is also discussed in the context of the P2P telepresence system. It is observed that the protocol can be used to elect the MC/MCU of CMTS for maximizing the DRDI of the system in bounded waiting time. The use of ZePoP for MC election in CMTS helps to relax assumption 1. The performance of ZePoP and GAncestor is compared with Static MC in terms of total traffic, hotness and composition time. Both the ZePoP and GAncestor significantly reduce the hotness values of the intermediate nodes but are still too high for processing all the streams with a high frame or bit rate. So, in the next chapter, the distribution of MPs is presented so that the hotness values of intermediate nodes are further reduced. The distribution of MPs is done by relaxing or removing assumptions 2 and 3. In the future, the focus will be given to improve and analyze the protocol in terms of message complexities in the first phase.

CHAPTER 5

Strategies for Distributed MCU

5.1 Introduction

This chapter is for phase2 of the P2P design of CMTS. It discusses two strategies to form the distributed MCU by relaxing assumptions 2 and 3. The first strategy aims to remove assumption 2, i.e., distribute the MPs satisfying the constraints on the resources of the peers but do not consider the communication cost among the distributed MPs. In the second method, an analytical model is presented, which suggests putting a ring of MPs on the SCDT without worrying about satisfying the node constraints. The ring placement method considers the communication cost among the MPs, thus removes assumption 3. In CHAPTER 3 and 4, the protocols create the balanced SCDT or DCDT by minimizing total hop count or mean delay from all nodes to the MC. They help to maximize the DRDI or the profit function. Both SCDT and DCDT are created, assuming each node has enough computational and communication power to deliver and combine all streams at the root (MC) of SCDT/DCDT within the given waiting time. It is observed that the nodes near the MC have too high hotness or load. In practice, each computing device has limited resources. So, the nodes with high hotness will induce excessive delays to deliver all streams to the MC. Moreover, the MC itself can handle a limited number of streams. So, the load distribution, especially near the vicinity of MC, is necessary, which is addressed in this chapter. In a distributed computing model, waiting time management is a more severe problem. Network delay is now quite manageable in the modern network. But computation delay added in each node becomes a significant consideration. So, an adaptive waiting time management scheme is also presented in this chapter to assign timers at MPs.

5.2 System Model

After each MC election, we have the SCDT or DCDT, where the root is the optimally placed MC. Now, how to reduce the computational load of this centrally overloaded MC? A simple principle is used. When the streams of the participants travel towards the MC, some MP nodes, if they have good computation power, can help and partially pre-mix some streams so that the communication and computation loads are reduced at the upper level of the SCDT. This distributed composition still must meet the timing constraints. The final mixing or composition is still done at the MC, and the combined stream is multicasted to all participants using SCDT.

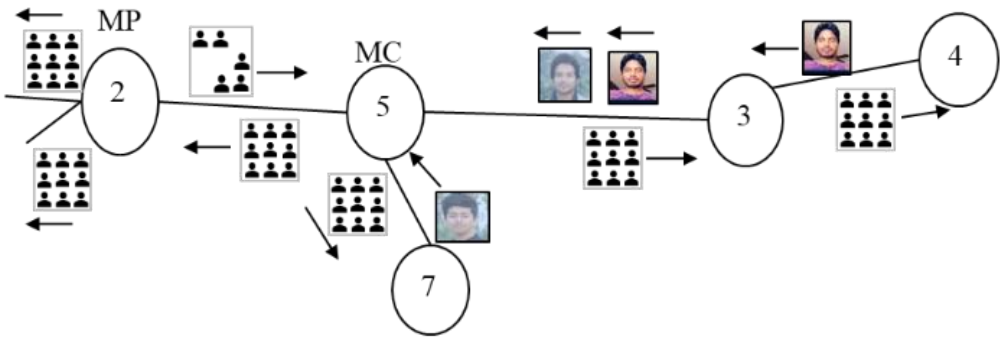


Figure 5.1 An example SCDT for CMTS with Distributed MCU

As an example, Figure **5.1** shows the SCDT of an MTS of 9 participants. The nodes 1, 6, 8, and 9 are not shown, but they are connected in two subtrees of node 2. Suppose each node sends a raw stream towards the MC. The MC receives the raw streams from nodes 3, 4, and 7 because the links they travel on have enough bandwidth to carry those raw streams. On the other hand, node 2 receives four raw streams from nodes 1, 6, 8, and 9. So, it has five raw streams, including its own stream. Suppose it doesn't have enough upload bandwidth to send all raw streams toward the MC. In other words, if node 2 sends all these raw streams to node 5, it will take a long time because of the low bandwidth on the link. So, node 2 decides to work as an MP. It mixes these streams, placing them at a related position in the grid-layout combined frame, and send the combined single stream to node 5. If node 2 doesn't have enough computing power for mixing all the streams, it may ask its children to send a partially combined stream, thus reducing computing load at node 2. The MC node also needs to process five streams (four raw and one combined stream) instead of nine. Thus, the distributed MCU is formed, and it should maximize the bounded time DRDI at the MC.

5.3 Distributed MCU by Satisfying Constraints

5.3.1 Problem Formulation

Suppose there are n participant peers in a CMTS where $\mathrm{n} \geq 2$. Their logical topology forms an undirected graph G = (V, E), where V is the set of all participants, and E is the set of edges among them. From G, there is an optimal SCDT rooted in the MC of the CMTS. Suppose each participant node i has upload bandwidth U_i, download bandwidth D_i, video mixing capacity B_i. The participant also has a profile weight in the conference w_i and a demand score v_i. In a CMTS session, each participant originates a raw multi-channel stream of conference rate *s* only if $v_i > 0$. These streams travel towards the MC using the SCDT. These streams must be partially processed by the nodes along the path to MC so that the system can maximize the DRDI at MC within the given waiting time. These additional processing nodes can be considered as the MPs of the MCU. So, we have to form a distributed MCU by selecting some MPs nodes in the SCDT rooted at MC to maximize the DRDI defined in (1.2) satisfying the following constraints,

i. at each peer i,

$$\boldsymbol{s \times n_r^i \leq D_i}, \tag{5.1}$$

$$\boldsymbol{s \times n_f^i \leq U_i} \tag{5.2}$$

ii. at each MP *j*, if the cost of boxing a raw video is *p*, boxing cost to put a boxed stream in the combined stream is *q*, then

$$\boldsymbol{r_j \times p + m_j \times q \leq B_k} \tag{5.3}$$

Where, n_r^i is the number of streams the node i receives and n_f^i is the number of streams it needs to forward during the conference. r_j is the number of raw video streams the node *j* needs to box and m_j is the number of already boxed streams it needs to put in the combined stream.

5.3.2 Solution Approach

After each MC election, we have the SCDT or DCDT, where the root is the optimally placed MC. The MC placement protocols help to increase the profit or DRDI by minimizing the path delays from all nodes to the MC. The formation of distributed MCU must further improve path delays, therefore the profit function. To form the distributed

MCU, the MPs can be placed in the SCDT, considering the computing capacity, bandwidth, and hotness information of the nodes. As defined earlier, the node hotness is the number of streams a node would send to its parent in the SCDT. So, the hotness is the load that needs to be distributed among other nodes. The leaf nodes have the hotness 1. So, the leaf nodes can start forwarding their hotness information to their parents using the HOTNESS message. Then the intermediate nodes calculate their hotness from the HOTNESS messages received from their children.

Table 5.1 The Symbols Used in the Solution

Symbols	**Description**
SCDT	The Stream Collection and Distribution Tree
U_i	The upload bandwidth of the node *i*
D_i	The download bandwidth of the node *i*
B_i	The mixing or boxing capacity of node *i*
N_i	The number of neighbors of node *i* in the SCDT
N_i^t	The number of children of node i in SCDT, i.e. $N_i = N_i^t + 1$
n	The number of participants in the session
n_r^i	The number of streams received by the node *i*
n_f^i	The number of streams forwarded by the node *i* to its parent in the SCDT
m_j	The number of already mixed streams
s	The conference rate
F_i	The forwarding hotness of node *i*
$f_{i,j}$	The hotness received by the node *i* from its child *j*
c_i^{max}	It is the maximum number of direct connections node *i* can make with other participants.

If a node has very high hotness compared to the bandwidth and the computing capacity, then it works as an MP and pushes down its load toward its children if required. This continues until each node has the hotness within its capacity, reducing the buffering at the upper level. The distribution of MPs reduces the total traffic in the SCDT. So, it is expected that the mean delays will be significantly minimized. Hence the DRDI will be maximized.

5.3.3 MCU Load Distribution Satisfying Constraints

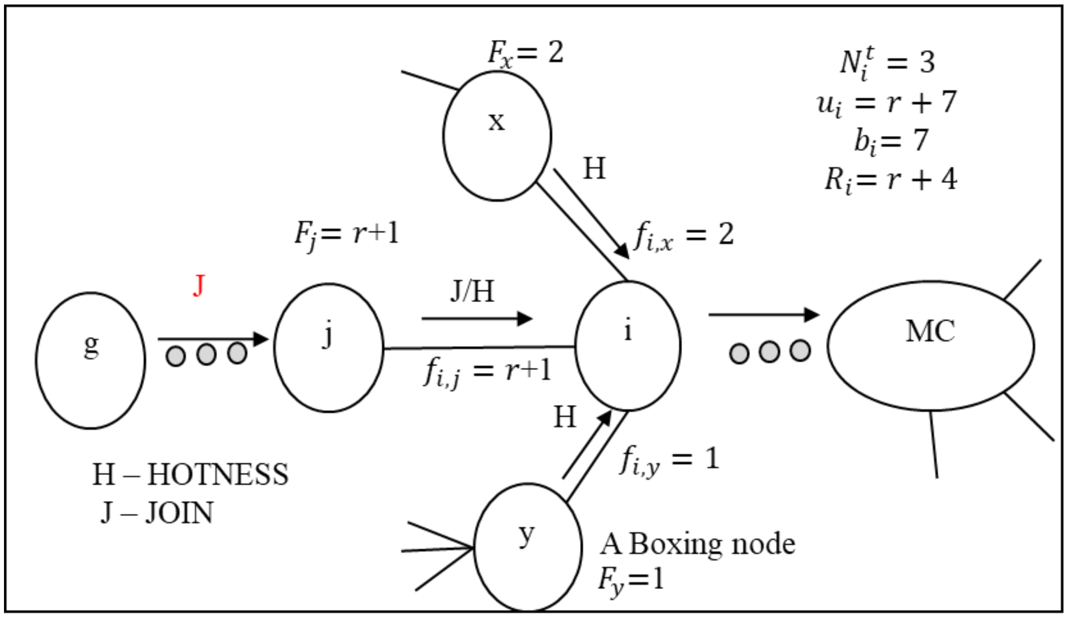


Figure 5.2 MCU operation distribution example.

Suppose the node *i* is the parent of node *j* in the SCDT, Figure 5.2. The node *j* has forwarding hotness F_j which is the number of raw or boxed streams the node *j* would send to node *i* during the telepresence session. The node *j* informs this hotness to *i* via

HOTNESS message. When the node *i* receives that message, it stores the hotness value contained in the message into the local branch variable $f_{i,j}$. If N_i^t is the number of children of node *i* in SCDT, then after receiving HOTNESS message from all children, the node *i* calculates its total hotness as follows:

$$\boldsymbol{R_i} = \sum_{\boldsymbol{j=I(i,k),1 \le k \le N_i^t}} \boldsymbol{f_{i,j}} + \boldsymbol{1} \tag{5.4}$$

Where the function *I (i, k)* returns the ID of kth child of *i*. If the number of descendants of node *i* in the SCDT is a_i then the actual hotness (number of raw streams node *i* would have) of the node is A_i = a_i + 1. So, $R_i \le A_i$. If $R_i < A_i$ then node *i* would receive at least one boxed stream of multiple streams. The HOTNESS message from *j* also contains a vector VF^j of stream flags. If $VF_k^j = 1$, it indicates that node *j* has a descendant node *k* in the SCDT tree. Each node also maintains this vector locally to locate individual streams in the combined stream. The node *i* creates its stream flag vector VF^i by setting $VF_i^i = 1$ and combining stream flag vectors from all children. The value of A_i would be the number of 1's in VF^i.

Now, based on the value of R_i and the resources, node *i* decides whether it has to work for the MC, i.e., box all R_i streams and send one combined stream to the parent. Suppose $d_i = \left\lfloor {}^{D_i}/_{S} \right\rfloor$, $u_i = \left\lfloor {}^{U_i}/_{S} \right\rfloor$ and $b_i = \left\lfloor {}^{B_i}/_{S} \right\rfloor$. So, d_i is the number streams the node *i* can receive without buffering, and u_i is the number of streams the node *i* can forward to its neighbors and b_i is the number of streams the node *i* can process without buffering. During a session of CMTS, the node *i* would multicast the final composite stream received from MC to all children. It can send all R_i streams to the parent without boxing if it has available upload bandwidth i.e. $R_i \leq (u_i - N_i^t)$. Otherwise, it has to box all R_i streams into one and send it to the parent. However, if $R_i > b_i$then *i* cannot combine them all. So, it distributes boxing load to some of its children by telling them to send one combined stream from each. If node *i* tells child *j* for boxing, it sets $f_{i,j} = 1$ and R_i is recalculated as $R_i = R_i - f_{i,j} + 1$. This is done until $R_i \leq\ b_i$. At this point F_i is set to 1 if it needs to work as a boxing point, otherwise $F_i = R_i$. Then F_i and VF^i are informed to the parent of *i* via a HOTNESS message. The node *i* also might receive instruction to work as a boxing point even if $R_i \leq (u_i - N_i^t)$ for lack of resources at its parent. Then it adjusts R_i if required as explained above so that $R_i \leq\ b_i$. Thus, some peers around the MC take responsibility for stream mixing, i.e., become MPs. It helps to reduce the hotness of MC and nodes around it significantly.

When a new node joins the telepresence session, it sends JOIN message towards the MC. Some nodes along the path of JOJN message need to recalculate the different local variables. The distribution of boxing load also might change. The JOIN message contains the forwarding hotness and vector VF of the node forwarding JOIN message. Suppose the ID of the new node is *g*, as shown in Figure 5.2. When a node *i* receives the JOIN message via node j, it performs the following calculations: $R_i = R_i - f_{i,j}$,$R_i = R_i + hotness(JOIN),\ f_{i,j} = hotness(JOIN), A_i = A_i + 1$. It also sets a video flag $VF_g^i = 1$, If *g* = *j* then node *g* is a new child of node i, so it increases N_i^t by 1. It also increases $I_{i,j}$ and $O_{i,p}$ by 1, where p is the parent of i. Now, based on the new value of R_i, node *i* decides if it has to work as a boxing point similarly as described for the HOTNESS message. The updated value of F_i and VF^i are forwarded to parent in JOIN message. A boxing node is not changed to non-boxing even if $R_i \leq (u_i - N_i^t)$ becomes true for new value of R_i to avoid many possible reconfigurations at upstream nodes. Similar calculations are performed when an existing node leaves the session. A boxing point is changed to non-boxing if all its descendants leave the CMTS session.

5.3.4 Example

Suppose, after MC election, a node *i* has three children x, y, and j, and $b_i = 7, u_i = r + 7$ and has enough download bandwidth. so $N_i^t = 3$, Figure 5.2. All of them send HOTNESS (H) message to node i. Node y is a boxing point. So, it will send a combined stream to i. Therefore, at node i, $f_{i,x} = 2$, $f_{i,j} = r$, $f_{i,y} = 1$, $R_i = r + 4$. As the node y send combined hotness 1, so $R_i < A_i$. Since available upload bandwidth at i, $u_i - N_i^t = r + 4$ which is equal to R_i, all streams can be sent to the parent node without boxing. So, $F_i = R_i$ and node *i* remains non-boxing. Now, suppose a new node g informs its joining by sending a JOIN(J) message toward MC. When node i receives the message, it recalculates $R_i = r + 5$, which is greater than the available upload bandwidth r+4. So, *i* becomes a boxing point and sends $F_i = 1$ toward its parent using the JOIN message. Now, if $r \leq 2$, then node *i* can box all streams by itself, since $R_i \leq b_i$. Otherwise, it has to inform node j to send only one combined stream. Then $R_i = 4$, and therefore $R_i \leq b_i$ would become true.

Theorem 5.1 *The MCU distribution algorithm guarantees to satisfy the bandwidth and computing capacity constraints.*

Proof: The MCU load distribution algorithm guarantees the delivery of all streams to the MC node by limiting the value of c_i^{max} which is the maximum number of direct connections node *i* can make with other participants. In other words, c_i^{max} is the maximum degree of the node *i* can have in the P2P overlay network. It is calculated based on the available resources of the node. The algorithm sets $c_i^{max} = \min(d_i, u_i, (b_i - 1))$, where $d_i = \lfloor {}^{D_i}/_{s} \rfloor$, $u_i = \lfloor {}^{U_i}/_{s} \rfloor$ and $b_i = \lfloor {}^{B_i}/_{s} \rfloor$. It means,

(i) If $c_i^{max} = d_i$ then $d_i \leq u_i$, $d_i < b_i$ and it is guaranteed that the peer *i* has available bandwidth to receive at least one stream of rate *s* from each of its neighbors. However, during a telepresence session, only links of the SCDT are utilized. A node *i* receives one composite stream from the parent and one or more streams from each child. If the number of neighbors of node *i* in SCDT is N_i then $(d_i - N_i)$ streams can be received from neighbors at multi-hop distance. As given in equation **(5.1)**, n_r^i is the total number of streams node *i* receives from all its neighbors in the SCDT. Then,

$$n_r^i = number\ of\ streams\ from\ children\ (R_i - 1) + number\ of\ streams\ from\ parent(1)$$

$$n_r^i = (R_i - 1) + 1$$

$$\boldsymbol{So, n_r^i = R_i} \qquad (5.5)$$

So, we can use n_r^i and R_i interchangeably in the calculations. The peer i satisfies constraints for both $D_i$and B_i if $n_r^i \leq c_i^{max}$, i.e $n_r^i \leq d_i$. Because $n_r^i \leq d_i$ is the same as the equation **(5.1)** and $d_i < b_i$. Therefore $n_r^i \leq b_i$ which also satisfy the equation **(5.2)** as $n_r^i = r_i + m_j$. If $n_r^i > c_i^{max}$, then there is at least one child who wants to send multiple streams (including forwarded streams from multi-hop neighbors) to node *i*. In this case, the node *i* adjusts R_i by informing some multiple streams senders to send one combined stream until $R_i \leq c_i^{max}, i.e\, n_r^i \leq c_i^{max}$. Now, if node i is the MC then it boxes and merges all streams and sends back the composite stream to all. If it is an intermediate node in the SCDT, then it boxes all R_i streams into one combined stream if the parent informed to do so for lack of resources upstream or it doesn't have enough bandwidth to forward all R_i streams. In the SCDT, each node *i* multicasts composite video from MCU to its N_i^t children. So, the available bandwidth for such node towards the parent is $(u_i - N_i^t)$. Therefore, node *i* can forward all the received streams towards MC if $R_i \leq (u_i - N_i^t)$, otherwise, one combined stream is forwarded as it has enough boxing capacity ($R_i \leq b_i$). Thus, limiting the number of connections based on download bandwidth, all constraints are satisfied, and each node guarantees to forward all received streams towards the MC using either it's available bandwidth or boxing capacity.

(ii) Now, if $c_i^{max} = (b_i - 1)$ then $(b_i - 1) \leq u$ and $(b_i - 1) \leq d_i$. So, it is guaranteed that node *i* can box or merge at least one stream from each of its neighbors. If the total intended number of streams from all neighbors $n_r^i \leq b_i$ then $r_i \leq d$ and decision on constraint u_i can be taken, as mentioned in case (1). Note that among n_r^i streams, one is the composite stream from the parent. It does consume the computing power, which can be used to mix the stream of the node *i* itself. If $n_r^i > b_i$ then first R_i has to be adjusted as described in case (1) to satisfy $R_i \leq d_i$ so that the download bandwidth constraint, $n_r^i \leq d_i$ is satisfied. Now, if node *i* has been informed by its parent to work as a boxing point or $R_i > (u_i - N_i^t)$ then R_i is adjusted again to make $R_i \leq b_i$ so that it can combine all streams it receives.

(iii) Similarly, if $c_i^{max} = u_i$, then it guarantees that the node *i* can send at least one stream to each of its neighbor (one stream toward MC and one composite stream to each of N_i^t children. However, $n_r^i \leq d_i$ always has to be true. If it's not, then R_i is adjusted to make it true. If $n_r^i > b_i$ then *i* works the same way as described in case (2) to make sure $n_r^i \leq b_i$ is satisfied when required.

We can see that d_i and b_i put a restriction on how many streams can be received from the neighbors and u_i restricts the number of streams that can be sent to neighbors. Thus, limiting the maximum number of connections of each node based on its available resources can guarantee the satisfaction of all constraints.

The constraint satisfaction enforces the distribution of MCU tasks or MPs. As the MPs combine multiple streams into one, so the total traffic is significantly reduced. Then the H_M can be redefined as follows,

$$\boldsymbol{H'_M} = \sum_{\boldsymbol{i\epsilon V, i\neq M}} \boldsymbol{F_{i,p_i}} \qquad (5.6)$$

Where the p_i is the parent of node i in the SCDT. At this point, the total traffic is more appropriate for H'_M than the total hop count. If there are one or more MPs in the SCDT, then $H'_M < H_M$ always true. So, distributed MPs always reduce the mean delay from all nodes to the MC by significantly reducing the traffic in the SCDT. However, the computation delays at MPs might increase some path delays causing to increase in the mean. But overall, DRDI maximization is expected. The effect of computational delays on the mean is analyzed in the later part of this chapter. This method does not consider the communication delays among the MPa, so the telepresence session's actual topology can infinitely grow if there is at least one node in the network capable of making connections with the new coming participant. So, we have to limit the size of the network based on the allowable waiting time at MC to receive all participants’ streams or maximize the DRDI. The waiting time management is discussed in the next section.

5.4 Wait Time Management

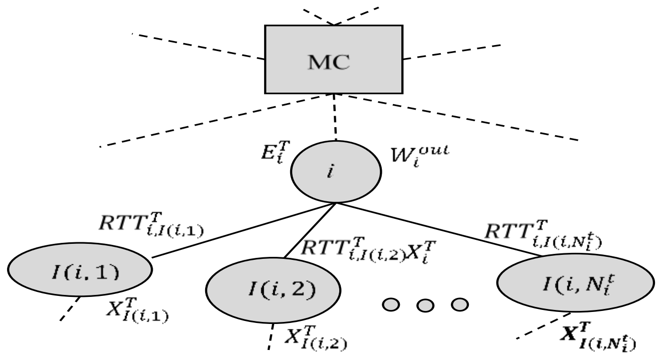


Figure 5.3 A part of SCDT

In a distributed computing model, the wait-time management is a more serious problem as significant processing delay is added. During a telepresence session, each non-boxing node forwards all streams from descendants to parents immediately. But each boxing point *i* waits to receive streams from all descendants until a timeout W_i^{out} expires, or it aggregates at least one video frame from each descendant node. The timer is started after each stream composition. In [81], several heuristics for timer management were presented, and simulation results showed that the heuristic with a fixed margin over cumulative delay (MCD) worked better in almost every situation. However, we cannot use MCD directly in our telepresence system because it is based on the accumulation of two-way delays. In the telepresence session, the wait time at a node would be proportional to the one-way delay from the furthest descendent to that node. Moreover, we need an adaptive scheme of timer management, which response with persistent variations of delays during the session. This helps to reduce both video composition time and video loss. The proposed adaptive timer management scheme is similar to MCD and based on the one-way delay, which consists of nodal delays and node to node propagation delays along the critical path as defined in [81]. The nodal delay includes queuing delay and the completion time of one iteration of the stream processing module. Suppose the node *i* in the SCDT has N_i^t children, Figure 5.3. It has nodal delay e_i^T measured at time *T*. In a multithreaded machine, the nodal delay can vary over time. So, we take the Exponential Weighted Moving Average (EWMA) of e_i^T as follows:

$$\boldsymbol{E_i^T = \beta E_i^{T-1} + (1-\beta) e_i^T, \quad 0 \le \beta \le 1} \tag{5.7}$$

Similarly, if $p_{k,i}^{T}$ is the propagation delay from child k to parent node i, then the EWMA value of propagation delay $P_{k,i}^{T}$ can be calculated as follows:

$$\boldsymbol{P_{k,i}^{T} = \gamma P_{k,i}^{T-1} + (1-\gamma) p_{k,i}^{T}, \quad 0 \leq \gamma \leq 1} \tag{5.8}$$

Where β and γ determine the importance given to the historical value of E_i^T and P_i^T respectively. So, the approximate delay the node *i* can expect before receiving stream from each descendant is,

$$\boldsymbol{X_i^T = Y_{max}^T + E_i^T} \tag{5.9}$$

$$\boldsymbol{Y_{max}^T = MAX\left(P_{k,i}^T + X_k^T\right): k = I(i,j), j \epsilon (1, 2, \cdots, N_i^t)} \tag{5.10}$$

Where function *I* maps *(i, j)→S*, i.e. *I (i, j)* returns ID of j^{th} child of node *i*. If the function *I* returns a leaf node, then $X_{leaf}^T = E_{leaf}^T$. Each node *k* updates E_k^T and $P_{k,i}^T$ in regular interval, recalculates X_k^T and sends it to the parent node. So, setting the timeout W_i^{out} based on X_i^T will be adaptive to any increased or decreased delays in the network. To have better control on timeout W_i^{out}, we consider the EWMA value of X_i^T which is calculated as follows,

$$\boldsymbol{W_i^T = \delta W_i^{T-1} + (1-\delta) X_i^T \ , \quad 0 \leq \delta \leq 1} \tag{5.11}$$

This also helps to avoid any spike of network delays on W_i^T. The parameters β, γ, δ can be used to set the speed of convergence with increased or decreased delay in the network. Now, the waiting time W_i^{out} is calculated as follows:

$$\boldsymbol{W_i^{out} = \xi W_i^T} \tag{5.12}$$

Where ξ is a constant and $1 \leq \xi < 2$. So, the constant adds a higher offset for nodes vicinity to the MC because W_i^T high if the node *i* near the MC.

5.5 Experimental Result

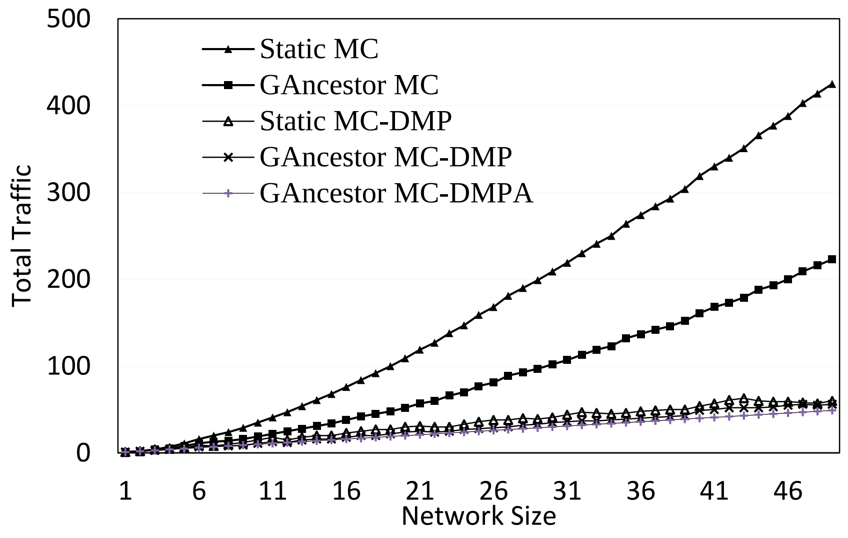
5.5.1 Experimental Setup

The proposed system is implemented as a distributed application with MPI, C++, and socket programming. All messaging is done via MPI, and only video transmission is done using TCP sockets. The system is simulated on a local cluster with n = 50. It is considered that each node i has enough download bandwidth because, in practice, the download bandwidth is much higher than the upload bandwidth. So, the constraint satisfaction only based on the upload bandwidth should be enough. The value of u_i is assigned randomly from the range [3, 5]. The maximum number of connections are allowed $c_i^{max} = 0.7u_i$. Then the boxing capacity count b_i is assigned to $c_i^{max} + 1$. α is always set to *2*. When the program was run on a machine using MPI on the node with ID 0 (assigned from MPI), nodes *1, 2, ⋯, n-1* sequentially joined the conference after every *10* seconds. For creating SCDT and electing the dynamic MC, the GAncestor protocol from CHAPTER 3 is used. Then the MPs are distributed as discussed in section 5.3. For the experiment, dummy streams of size 1KB with header 23 bytes' header were transmitted from the participants. The composite dummy video of the same size was generated by simply taking an equal portion of everyone's video packet. For comparative analysis, we run the system for five settings (1) Static MC with no constraints applied, referred to as

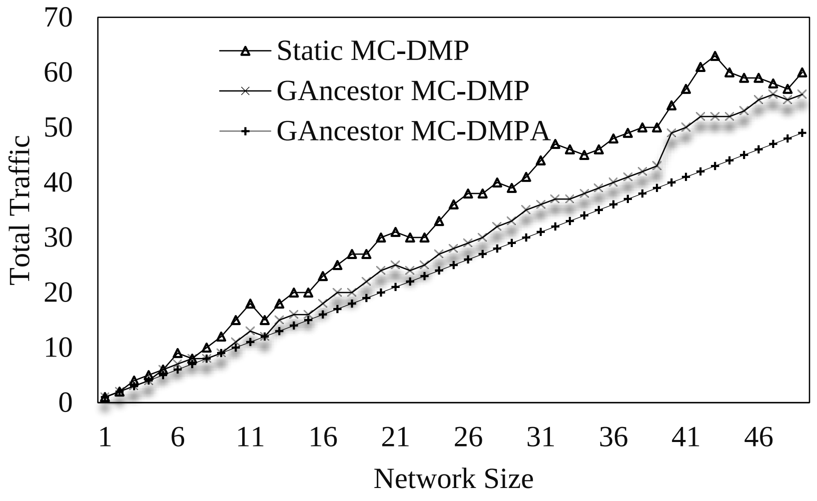
Static MC (2) Dynamic MC with no constraints applied, referred to as GAncestor MC (3) Static MC with bandwidth and boxing capacity constraints applied. So, this setting is for distributed MPs and referred to as Static MC-DMP (4) Dynamic MC with bandwidth and boxing capacity constraints applied and referred to as GAncestor MC-DMP (5) Dynamic MC and making all nodes (except leaf) mixing point or MPs. So, the MCU distribution method discussed above is not used. Each node except the leaf node is assigned to collect the streams of its descendant, combine them and send the combined stream towards the parent. This setting is referred as GAncestor MC-DMPA. Each boxing node (MC and MPs) uses the adaptive waiting management scheme to set the waiting time to receive streams of its descendants. In the subsequent sections, the recorded results for these system settings are discussed.

5.5.2 Comparison on Total Traffic, Hotness and Composition Time

For comparison, the system is run for all the five settings by forcing the application to create the same network topology incrementally for up to 50 nodes. The total traffic is recorded for each system setting for different network sizes.



(a)



(b)

Figure 5.4 Effect on total traffic as network size increases

Figure **5.4** compares the different settings based on total traffic. In (a) total traffic is compared for all five settings. As the number of nodes increases in the system, the total traffic increases for all the system settings. We have already observed in CHAPTER 3 that the total traffic of GAncestor MC is always lower than the Static MC case. The system settings 3, 4, 5 are for distributed MCU. As we can see, the total traffic in distributed MCU cases is significantly lower than the centralized Static MC and the GAncestor MC cases. For better understanding, a separate comparison between the distributed MCU settings are shown in Figure **5.4** (b). As a new node joins the conference (network size increases), only a single traffic unit is added for GAncestor MC-DMPA because each link carries only a single stream or traffic (either combined or raw). Therefore, the related graph is a straight line. For Static MC-DMP and GAncestor MC-DMP, few random links will carry multiple streams. So, their total traffic lines are above the line of GAncestor MC-DMPA. Because of the better position of MC, the traffic line for GAncestor MC-DMP is lower than the line for Static MC-DMP. The major reduction of the total traffic of the system with distributed MCU should improve the overall delay in the system.

As discussed earlier, in distributed MCU settings (3, 4, and 5), the MPs are distributed based on the upload bandwidth because the download bandwidth is usually high compared to the upload bandwidth. So, the hotness of the intermediate nodes would be limited by their upload bandwidth, i.e., the number of connections they make in the topology, which is between 3 to 5, according to the experimental setup. More specifically, the hotness of an intermediate node will be closely related to the number of children it has in SCDT plus one.

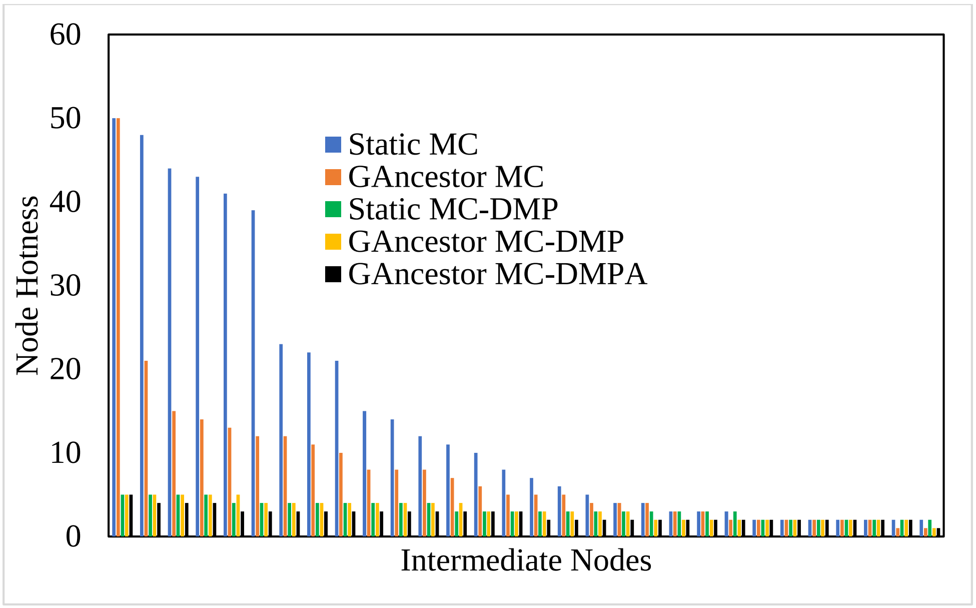


Figure 5.5 Comparison of hotness of intermediate nodes(n=50)

For comparison based on the hotness of intermediate nodes, the CMTS application is run on the same logical topology of 50 nodes for all the system settings. The comparison is shown in Figure **5.5**. The bar chat shows the hotness values of intermediate nodes in decreasing order along the x-axis for all the system settings. As we can see, the GAncestor MC significantly reduces the hotness of the intermediate nodes electing the optimal MC node (we also observed this in CHAPTER 3). All the distributed system settings (3, 4, and 5), distribute loads of the MC (optimally placed by the GAncestor protocol) as well as the intermediate nodes of the SCDT. Thus, all these schemes bring down the hotness values to the tolerable level of the intermediate nodes with respect to their upload bandwidth. This also helps to reduce the load on the related links. The hotness values of Static MC-DMP and GAncestor MC-DMP are similar, but GAncestor MC-DMPA seems to yield lower hotness for the intermediate node of the SCDT. Therefore, it is expected that the frame time. i.e., the composition time will be significantly lower for all the distributed MCU settings compared to the centralized Static MC and GAncestor MC settings. But GAncestor MC-DMPA will perform better than all other settings,

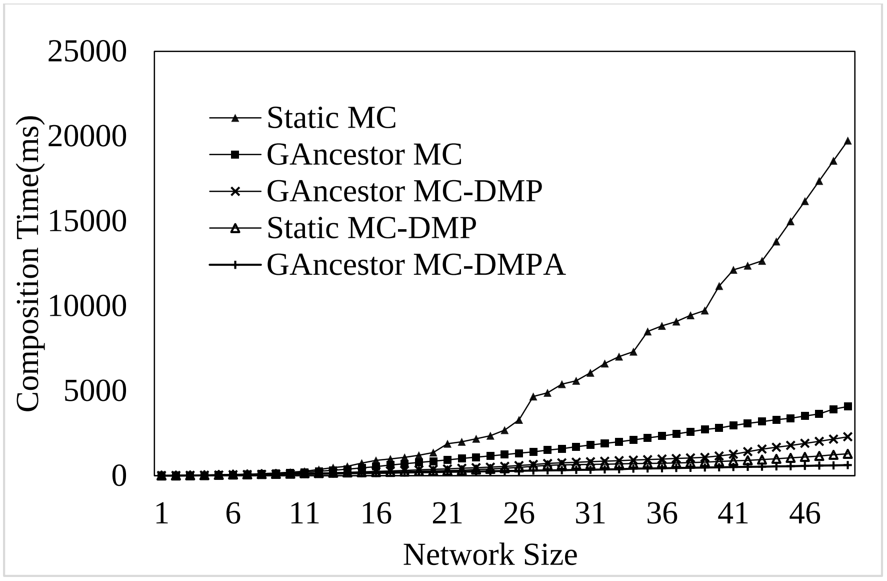


Figure 5.6 Comparison of composition time

For comparing the composition time, the application is run for all the five system settings with a fixed large waiting time at each boxing node, including the MC. The large waiting time ensures that the MC can collect the steams of all participants before generating the packet for the composite stream. During the CMTS session, the MC node records the average stream composition time for different network sizes. Figure 5.6 shows the cumulative values of such composition time for all the system settings. It shows that the Static MC always takes a long time to collect the streams from all participants. The reasons are the worst position of the MC node and the high load (hotness) at the links and the nodes near the MC node. Because of the better position of MC, GAncestor MC spends significantly low composition time. The composition time is further low for all the distributed MC cases (3, 4, and 5) compared to Static MC and GAncestor cases. It is very low for GAncestor MC - DMPA because of the lowest traffic in the SCDT. However, it is found that the composition time is higher for GAncestor MC-DMP compared to the Static MC-DMP. This might be for the cumulative effect of the position of boxing points and the physical location of the MC node. Because the GAncestor uses the hop-count for MC placement, not the actual latency of the links. Another reason we found is the number of required boxing points was higher, 27, for Static MC-DMP to satisfy all constraints. For GAncestor MC-DMP, it was 20. This can be a valid reason because GAncestor MC-DMPA gives the lowest composition time with many boxing points (all the intermediate nodes).

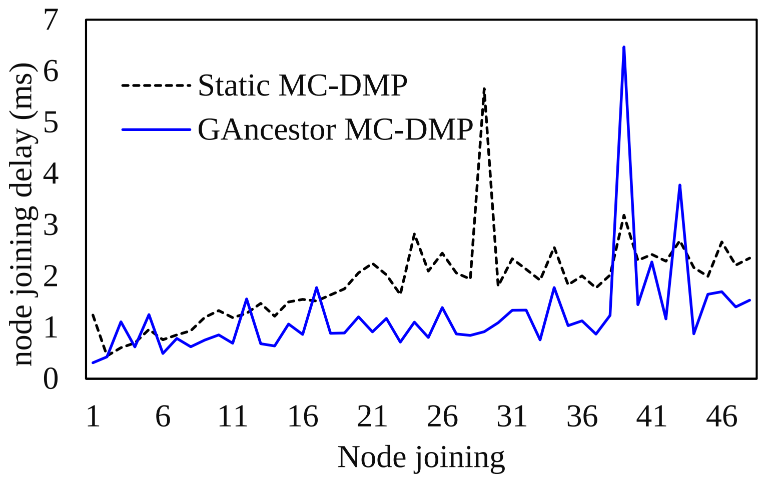


Figure 5.7 Comparison of node joining delays

It is important that the system allows new nodes to join the CMTS session as soon as possible, especially for user satisfaction. It is also observed that the dynamic MC placement with distributed MPs, i.e., GAncestor MC-DMP, can significantly reduce the delay for joining a new node. Figure 5.7 shows that the delays for a new node joining are higher for static MC-DMP compared to GAncestor MC-DMP. The locations of MCs for both GAncestor MC-DMP and GAncestor MC-DMPA cases are the same, so node joining delays are expected to be the same. Therefore, we have shown only one of them in the plot.

5.5.3 Loss-Profit Analysis

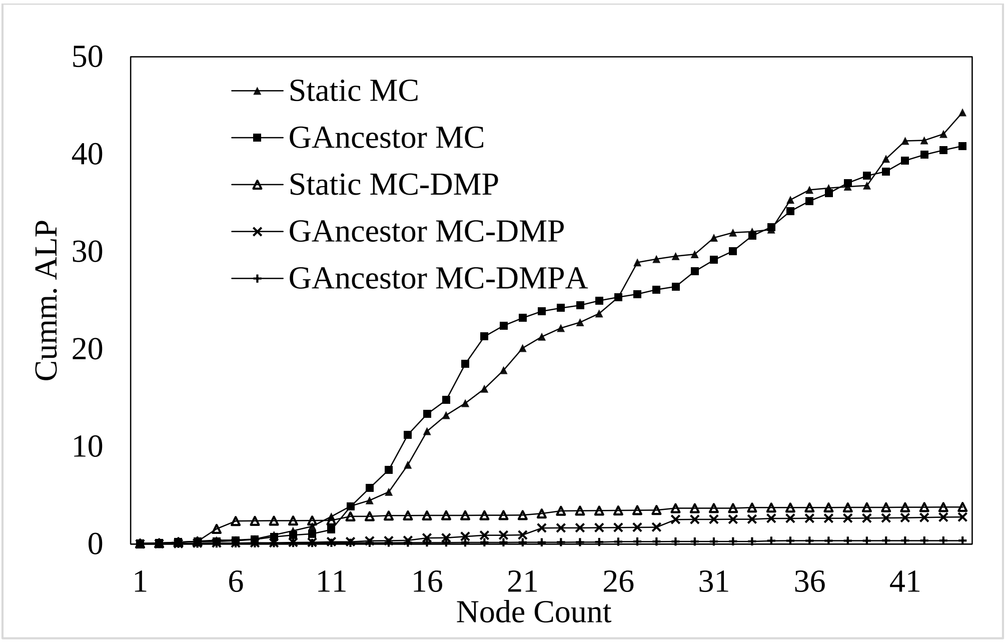


Figure 5.8 Comparison of Cumulative ALP

The metrics so far discussed are based on the large fixed waiting time at the boxing points. It is observed that the distributed MCU setups perform significantly better compared to the centralized single MC settings. Now, we analyze the performance of the five system settings based on a new metric called Average Loss Percent (ALP). Suppose in a CMTS session of *n* participants, the MC and MP nodes wait *W* milliseconds to receive the streams from their descendants. The value of W is calculated using the adaptive wait time management scheme. An MP node combines the frames of the received streams into one and forwards it towards the MC. The ALP can be defined as the percentage of participants who cannot deliver their streams to the MC within the MC's waiting time, i.e., miss the composition. Since the ALP is an average value, it is calculated over a certain time $\bar{T}$. Suppose the MC generates K_n composite frames over time $\bar{T}$. Then the ALP of the network is defined as shown below,

$$\boldsymbol{ALP_n = \frac{\sum_{i=1}^{K_n}\left\{\frac{(n-n_i')}{n}\times 100\right\}}{K_n}\times 100} \tag{5.13}$$

Where, n_i' is the number of participants that can deliver their streams to the MC during i^{th} composition before the waiting time expires. Figure 5.8 shows the cumulative value of ALP for all five settings. The centralized Static MC and GAncestor MC cases are again worst compared to the distributed MCU settings. The reason is the high hotness near the MC. It can be observed that the GAncestor MC-DMPA again performs better compared to GAncestor MC-DMP and static MC-DMP cases. So, it proves that high traffic and hotness within the network increases the ALP, i.e., increases the percentage of frame loss during stream composition. The high loss means low profit. So, for maximizing profit, more boxing points seem better. However, the wait time, propagation delays, and nodal delays at the MPs have a significant impact on the metrics we have discussed. Moreover, the communication cost from the MC to all MPs will be high if the number of MPs is very high. The impact of nodal, propagation delays, the position of MPs is analyzed in detail in the latter part of the chapter. So, far it is found that the more MPs and less traffic can significantly improve the mean delays in the system, thus can improve the DRDI of the system.

5.5.4 Effect of Adaptive Waiting Time

If a node is slow because of very low resources, then the topology formation technique and the MC election algorithm keeps these nodes near the leaf. However, a node may still become slow during telepresence sessions for many reasons. The locations of the slow nodes are essential. A slow node near MC has a significant impact on the system performance than the slow node near the leaves. The adaptive wait time management scheme can play a significant role in addressing the speed changes of these nodes and improve the overall performance of the system. For testing the effectiveness of the adaptive waiting time scheme, the application is run with GAncestor MC-DMP setting on 50 nodes topology. During different sessions, some nodes are forcefully slowed down at various locations (near MC and near leaf) for different durations, and the response of the MC node to adjust the waiting time is observed.

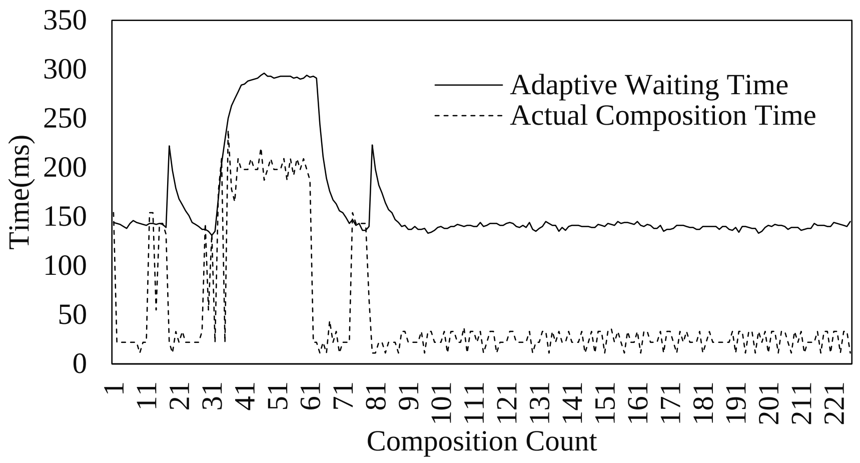
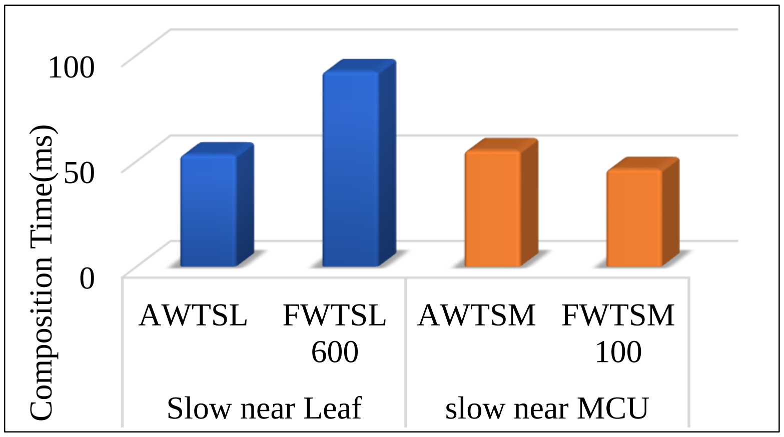
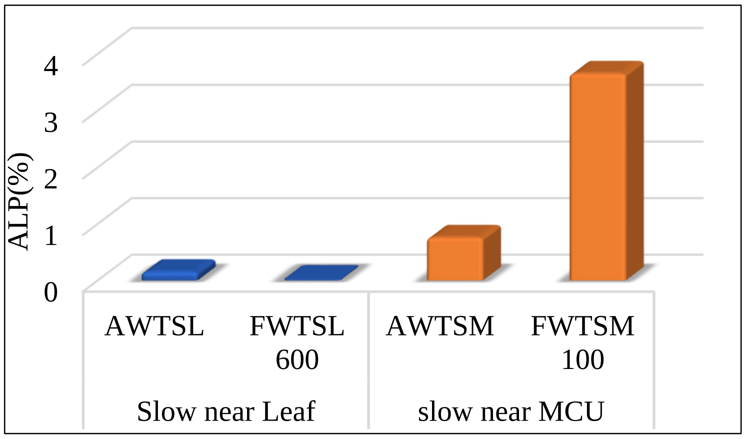


Figure 5.9 Adaptive waiting time



(a)



(b)

Figure 5.10 Average loss and delay comparison for adaptive waiting time scheme

Figure 5.9 shows the MC's adaptation of waiting time when a leaf node slows down for **short-long-short** durations. The same values for β, γ, and δ are used, and it is 0.8. Therefore, the MC responds a little slow by taking an average of the new delay of the critical path. It helps to avoid reacting immediately to a sudden high spike in path delay but adjust the waiting time for a persistent change of the delays.

The effect of the adaptive waiting time scheme is compared further with the schemes with the fixed large and small waiting time. The comparison is made based on four settings of waiting time when the system is run using GAncestor MC-DMP on 50 nodes session. These settings are (1) AWTSL - Adaptive Wait Time with slow node near Leaf (2) FWTSL 600 - 600ms Fixed Wait Time and slow node near Leaf (3) AWTSM - Adaptive Wait Time with slow node near MC (4) FWTSM 100 – 100ms Fixed Wait Time and slow node near MC. Figure 5.10 shows the comparison of (a) composition time and (b) average loss of these four settings. The result for FWTSM 600 and FWTSL 100 are not shown because the intention was to show only how a slow leaf node can increase overall delay at MC for high fixed waiting time (FWTSL 600) and how the Fixed short waiting time (FWTSM 100) can cause high loss of streams if a node near the MC becomes slow. FWTSM 600 would have a similar effect on delay as FWTSL 600, and FWTSL 100 would show a small loss (only the stream of the slow node and the node behind it) of streams. In (a), FWTSL 600 indicates that the maximum value of the average composition time for the experiment is 90ms. So, FWTSL 600 can guarantee to receive all streams within the waiting time of 600ms, which is the maximum DRDI. However, 100ms waiting time also should be enough for receiving the maximum DRDI. Therefore, FWTSM 100 is also considered for comparison with the adaptive waiting time performance. FWTSM 100 is expected to give low composition time and be enough to receive all streams. We can see that the delays for adaptive schemes (both AWTSL and AWTSM) are very close to the delay for FWTSM 100 and much lower than the delay for FWTSL 600. They are a little high because the adaptive schemes adjust with some long term high delays that are discarded by FWTSL 100. The adaptive scheme ensures overall low delay by avoiding waiting for a single slow leaf node. This increases very little high average loss compared to FWTSL 600, Figure 5.10(b). The FWTSL 600 is expected to ensure maximum DRDI (almost zero loss) but with high composition time. On the other hand, when the slow node is near the MCU, there is a high possibility of frame loss with the short wait time (FWTSM 100) and the result also confirms that. The adaptive scheme again ensures a significantly lower loss percent by adapting the waiting time at the boxing points. Thus, the adaptive setting of waiting time can keep both composition time and ALP or frame loss lower irrespective of the slow node's location. Therefore, it can further maximize the profit or DRDI of the telepresence system with distributed MCU.

The ALP can be considered as the measure of profit loss. So, the Profit Collection Efficiency(PCE) can be defined as follows,

$$PCE(n) = \frac{100 - \frac{ALP_n}{100}}{\overline{D(n)}}$$

Where $\overline{D(n)}$, is the average composition time when the number of nodes in the CMTS session is *n*.

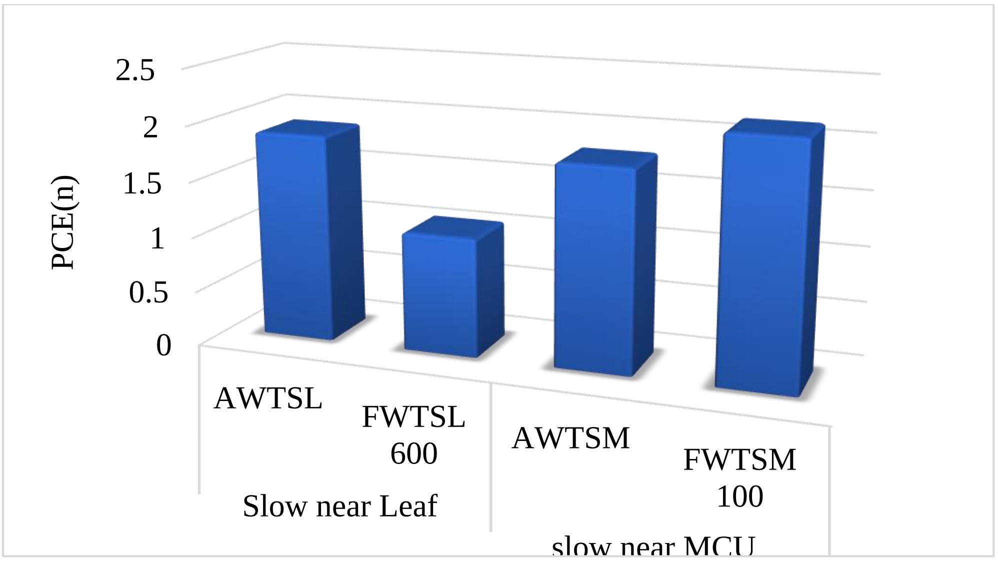


Figure 5.11 Comparison of waiting time settings based on PCE

Figure 5.11 shows the PCE for different waiting time settings calculated from Figure 5.10. As we can observe, the adaptive waiting time management scheme can always ensure better PCE values. However, the adaptive timer depends on the path delays in the network. Some slow nodes can significantly increase the overall composition time. Also, as the number of nodes increases, the adaptive timer will increase, and it should not be allowed to grow infinitely. Because it will decrease the PCE of the system. For CMTS, a fixed and reasonably small waiting time must be provided at the MC so that the system can guarantee high PCE when required. In that consideration, FWTSM 100 is best because it gives higher PCE. In the next chapter, we present an optimal timer scheme that uses a given fixed waiting time to assign optimal waiting time to all the MPs to maximize the overall profit and PCE.

5.6 Ring Placement Method for Distributed MCU

In this section, a ring placement method is discussed to form the distributed MCU, where all the MPs are placed at a fixed distance from the MC. Thus, the MPs form a ring around the MC. The method considers the synchronization cost among the MPs to place the ring, thus removes assumption 3. The method works based on the delay modeling on the system model of Distributed MCU.

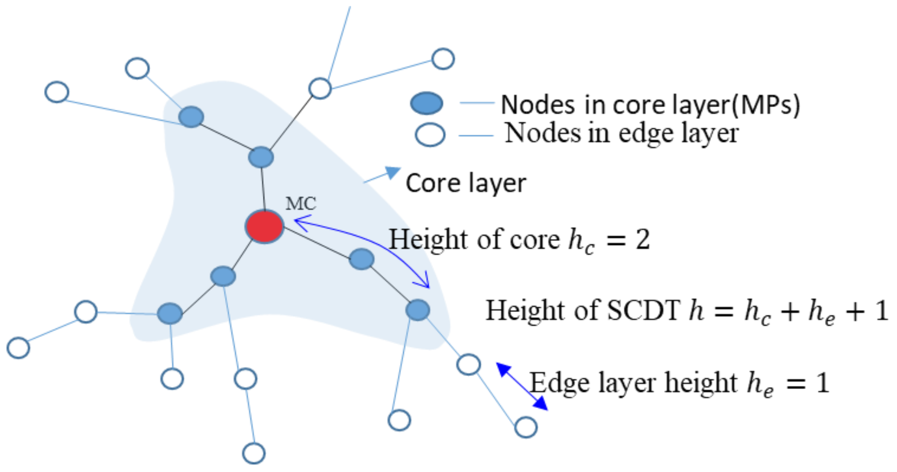


Figure 5.12 The SCDT with two layers of nodes

In the distributed MCU model, after placing the MPs around MC, the SCDT has two layers of participant nodes. The nodes working as MPs, along with the MC, create the core layer and other nodes sit in the edge layer, as shown in Figure **5.12**. The nodes on the core layer collect video streams or frames from all participants and generate a composite stream within a given waiting time *W*. The *W* is also considered as the frame interval, or it defines the rate of the composite stream. The final composition is done at the MC. Now, suppose the size of the core layer h_c which can be defined as the average or maximum distance of MP node(s) from the MC. The height of the edge layer is h_e. So, the branch length or height from a leaf node to MC is h= $h_c + h_e + 1$. Now if $h_c = 0$, then the MC must process everything. It means the CMTS session is based on a single MCU that has been proved to worst for CMTS. If $h_c > 0$, then the session is based on distributed MCU and has one or more MPs in the SCDT. But the large value of h_c will increase the number of MPs in the system and therefore, the synchronization delays and the related issues will also increase. So, the ring placement method aims to estimate the optimal value of h_c so that the overall path delays to MC from all other nodes are minimized, i.e., the DRDI is maximized. After finding the optimal h_c, all the nodes at distance h_c will be assigned to be the MPs, i.e., the ring of MPs is formed on the SCDT.

5.6.1 Delay Modeling for Distributed MCU:

Suppose $d_{x,y}$ is a point to point delay between two neighbors x and y in the SCDT. The path delay from node i to MC is D_i and the set of nodes along the path is $S = \{0, 2, 3, \dots h\}$ where the first node is i (one of the leaf nodes in SCDT), and the h^{th} node is MC. Then

$$D_i = \sum_{x=0}^{h-1} d_{x,x+1}$$

$$= \sum_{\boldsymbol{x=0}}^{\boldsymbol{h-1}} \left(\boldsymbol{b_x} + \boldsymbol{l_{x,x+1}}\right) \tag{5.14}$$

Where b_x is the stream boxing time and $l_{x,x+1}$ is the link latency between two neighbors x and $x+1$. Now, suppose the number of streams the node x can forward to the parent x+1 is r_x and it is the number of nodes in the subtree rooted at x. The number of children of x in SCDT is N_x^t, the size of the video frame F, boxing capacity of x is C_x, upload bandwidth U_x and ρ_x is the node status indicator. If x is in the core, then $\rho_x = 1$, and 0 otherwise. Then

$$\boldsymbol{b_x = \frac{r_x \times F \times \rho_x}{C_x}} \tag{5.15}$$

$$\boldsymbol{l_{x,x+1} = \frac{r_x^{(1-\rho_x)} \times F}{U_x} + \frac{N_x^t \times F}{U_x}} \tag{5.16}$$

So, path delay becomes

$$\boldsymbol{D_i = F \sum_{x=0}^{h-1} \left(\frac{r_x \times \rho_x}{C_x} + \frac{r_x^{(1-\rho_x)}}{U_x} + \frac{N_x}{U_x} \right)} \tag{5.17}$$

For a stream or frame to be part of a composite stream, it must reach the MC within a given waiting time W (ignoring the time required to process the last frame before *W* expires). To receive and process all participants within the time *W*, the maximum path delay plus final processing time at MC must be less than or equal to W. That is the composition time for Distributed MCU is,

$$\boldsymbol{D_{comp}^{DMCU} = maximum\ branch\ delay + time\ for\ final\ compostion} \tag{5.18}$$

But $D_{comp}^{DMCU} \leq W$ must be true for a composition with all participants' streams. From the diagram, the branch delay has two parts. One is from the core layer; another one is from the edge layer. The time at MC can also be part of the core layer. So, the equation 5.18) can be rewritten as follows,

$$\boldsymbol{D_{comp}^{DMCU} = D_{edge} + D_{core}} \tag{5.19}$$

For simplicity, let's consider the average number of children for each node in the SCDT is $Avg(N_x^t) = d$. In the edge layer, $\rho_x = 0$, i.e. zero computational cost. So, from (5.17),

$$\boldsymbol{D_{edge} = F\sum_{x=0}^{h_e}\left(\frac{r_x}{U_x}+\frac{d}{U_x}\right)} \tag{5.20}$$

For a tree with branching factor *d* and height *h*, the total number of nodes in the subtree rooted at x is,

$$\boldsymbol{r_x = \frac{d^{h+1}-1}{d-1}} \tag{5.21}$$

In the core layer, $\rho_x = 1$. Therefore, the D_{core} will be the summation of computational delay(D_{core}^{comp}) and communication delay(D_{core}^{comm}). The communication delay would be,

$$\boldsymbol{D_{core}^{comm} = F\left\{\sum_{x=h_e+1}^{h_e+h_c}\left(\frac{1}{U_x}+\frac{d}{U_x}\right)+\frac{\beta r_{h_c}}{U}\right\}} \tag{5.22}$$

The first part with the sum is the cost of forwarding a single combined stream towards MC and delivering the composite stream from MC to the children. The summation is for every core node along the path. The second part is for the communication cost required from MC to all nodes in the core because the MC needs to communicate with the MPs. In this part, average bandwidth in core $U = Avg(U_x)$, where x is a node in the core.

The message size for communication is considered as β fraction of a raw stream or frame. Now the computational cost at the core is,

$$\boldsymbol{D_{core}^{comp} = F\left\{\frac{r_{h_e+1}}{C_{h_e+1}} + \sum_{x=h_e+2}^{h_e+h_c+1\ or\ h} \left(\frac{1}{C_x} + \frac{\alpha d r_{x-1}}{C_x}\right)\right\}} \qquad (5.23)$$

The first node (at $h_e + 1$) in the core (connector to the edge layer) is responsible for processing all streams coming from the edge layer, and other nodes process their streams as well as merge the partially combined streams received from the children. Here $\alpha = F'/F$ where F' is the size of a boxed stream. So, after combining and reorganizing different parts of D_{core} and D_{edge},

$$\boldsymbol{D_{comp}^{DMCU} = F\left\{\left(\sum_{x=0}^{h_e} \frac{r_x}{U_x} + \sum_{x=h_e+1}^{h_e+h_c} \frac{1}{U_x} + \sum_{x=0}^{h-1} \frac{d}{U_x} + \frac{\beta r_{h_c}}{U}\right) + \left(\frac{r_{h_e+1}}{C_{h_e+1}} + \sum_{x=h_e+2}^{h_e+h_c+1\ or\ h} \left(\frac{1}{C_x} + \frac{\alpha d r_{x-1}}{C_x}\right)\right)\right\}} \qquad (5.24)$$

To observe the nature of the delay function above, we plot its normalized values against h_c with h=24 for different combinations of values of U_x and C_x keeping $\alpha, \beta \epsilon [0.1, 1.0]$.

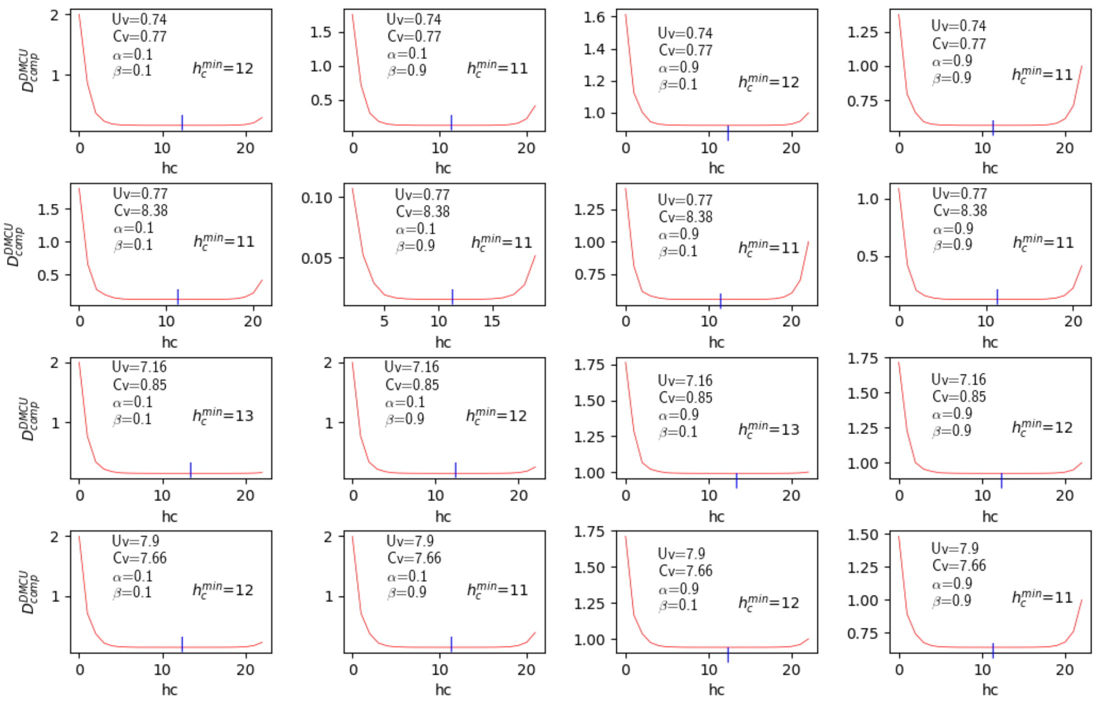


Figure 5.13 $\boldsymbol{D_{comp}^{DMCU}}$ observation for different values of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{U_v}$, and $\boldsymbol{C_v}$

Figure 5.13 shows 16 plots for 16 sets of values of these parameters. We can observe that as the value of h_c increase, the D_{comp}^{DMCU} decreases exponentially most of the cases until $h_c = h/2$, where h is the depth of the tree. However, it is not a monotonically decreasing function. It does start rising based on the value of α and β. The nature of the terms with β is also exponential because as h_c increases, the number of nodes in the core also increases exponentially. Similarly, the term with α is exponential. So, on the plot of D_{comp}^{DMCU} there must be a h_c^{min} where the delay is minimum. We show below how it is computed. The computed minimum point is marked with a blue vertical line on the x-axis.

5.6.2 Determining the h_c^{min}

From the graph, h_c^{min} is very close $\frac{h}{2}$. However, we have observed that it also moves away from the mid position, especially when $\frac{\beta}{U}$ is too small or too large. In all the figures, a significant portion looks flat. So, we zoom in on the middle position of the graph by plotting only the values near the minimum delay. It is observed that the nature of the graph is the same, i.e., exponential decrement at the beginning and then exponential growth after the minimum delay position. So, in this section, we discuss finding the h_c^{min}. The value will allow the system to automatically place the MPs without going through the complex MP distribution process.

For finding h_c^{min}, let us proceed to solve the equation (5.24) for h_e (or h_c). We can rewrite the equation as follows, replacing computing capacities C_x with average C and U_x with average bandwidth U.

$$D_{comp}^{DMCU} = F\left\{\left(\frac{1}{U(d-1)}\sum_{x=0}^{h_e}(d^{x+1}-1) + \frac{h_c}{U} + \frac{dh}{U} + \frac{\beta(d^{h_c+1}-1)}{U(d-1)}\right)\right.$$
$$\left. + \left(\frac{r_{h_e+1}}{C} + \sum_{x=h_e+2}^{h}(\frac{1}{C} + \frac{\alpha d(d^x-1)}{C(d-1)})\right)\right\}$$

Or

$$D_{comp}^{DMCU} = F\left\{\left(\frac{1}{U(d-1)}\left(\frac{(d^{h_e+2}-1}{d-1} - \ h_e\right) + \frac{h_c}{U} \ + \frac{dh}{U} + \frac{\beta(d^{h_c+1}-1)}{U(d-1)} \ \right)\right.$$

$$+\left(\frac{d^{h_e+2}-1}{C(d-1)} + \frac{h_c-1}{C}\right.$$

$$\left.\left.+\frac{d\alpha}{C(d-1)}\left(\frac{d^{h+1}-1}{d-1} - \frac{d^{h_e+2}-1}{d-1} - h + h_e + 2\right)\right)\right\}$$

Or

$$D_{comp}^{DMCU} = F\left\{\left(\frac{d^{h_e+2}-1}{U(d-1)^2} - \frac{h_e}{U(d-1)} + \frac{h-h_e-1}{U} \ + \frac{dh}{U} + \frac{\beta(d^{h-h_e}-1)}{U(d-1)} \ \right)\right.$$

$$+\left(\frac{d^{h_e+2}-1}{C(d-1)} + \frac{h-h_e-2}{C}\right.$$

$$\left.\left.+\frac{d\alpha}{C(d-1)}\left(\frac{d^{h+1}-1}{d-1} - \frac{d^{h_e+2}-1}{d-1} - h + h_e + 2\right)\right)\right\}$$

Now D_{comp}^{DMCU} is minimum when $\frac{dy}{dh_e} = 0$, where $y = D_{comp}^{DMCU}$

Or,

$$F\left\{\left(\frac{\ln(d)\, d^{h_e+2}}{U(d-1)^2} - \frac{1}{U(d-1)} - \frac{1}{U} \ + 0 - \frac{\beta \ln(d)\, d^{h-h_e}}{U(d-1)} \ \right)\right.$$

$$\left.+\left(\frac{\ln(d)\, d^{h_e+2}}{C(d-1)} - \frac{1}{C} + \frac{d\alpha}{C(d-1)}\left(0 - \frac{\ln(d)\, d^{h_e+2}}{(d-1)} + 1\right)\right)\right\} = 0$$

Or,

$$\left(\frac{\mathrm{Ln}(d)}{U(d-1)^2}+\frac{\ln(d)}{C(d-1)}-\frac{d\alpha\ln(d)}{C(d-1)^2}\ \right)d^{h_e+2}-\frac{\beta\ln(d)\,d^h}{U(d-1)d^{h_e}}$$

$$=\frac{1}{U(d-1)}+\frac{1}{U}\ +\frac{1}{C}-\frac{d\alpha}{C(d-1)}$$

Or,

$$\left(\frac{\mathrm{Cln}(d)+\ \mathrm{Uln}(d)(d-1)-d\alpha U\ln(d)}{UC(d-1)^2}\right)d^{h_e+2}-\frac{\beta\ln(d)\,d^h}{U(d-1)d^{h_e}}$$

$$=\frac{C+C(d-1)+U(d-1)-d\alpha U}{UC(d-1)}$$

$$(\mathrm{Cln}(d)+\ \mathrm{Uln}(d)(d-1)-d\alpha U\ln(d))d^{h_e+2}-\frac{\beta C\ln(d)(d-1)\,d^h}{d^{h_e}}$$

$$=(d-1)(\ C+C(d-1)+U(d-1)-d\alpha U)$$

Or,

$$\boldsymbol{Xd^2d^{2h_e}-Pd^{h_e}-Q=0} \qquad (5.25)$$

Where $X=\mathrm{Cln}(d)+\ \mathrm{Uln}(d)(d-1)-d\alpha U\ln(d)$, $Q=\ \beta C\ln(d)(d-1)\,d^h$ and $P=(d-1)(\ C+C(d-1)+U(d-1)-d\alpha U)$. Now solving equation (5.25) ,

$$d^{h_e}=\ \frac{P\pm\sqrt{P^2+4Xd^2Q}}{2Xd^2}$$

So,

$$\boldsymbol{h_e = \log_d\left(\frac{P + \sqrt{P^2 + 4Xd^2Q}}{2Xd^2}\right)} \tag{5.26}$$

We remove (-) from the formula because the term inside root square is always higher than P^2 and the whole value in the log would be negative. Therefore,

$$\boldsymbol{h_c^{min} = h - \log_d\left(\frac{P + \sqrt{P^2 + 4Xd^2Q}}{2Xd^2}\right) - 1} \tag{5.27}$$

So, equation (5.27) gives us the value of h_c where the maximum branch delay of SCDT is minimum. Table 5.2 shows some example calculation of h_c^{min} using the equation (5.27). The calculated values are like the values shown in the diagram for h=24 for similar parameter settings. But it is also observed that if the value β is very small(last row in the table) then the h_c^{min} can move far away from the $\frac{h}{2}$. Because β small means the communication overhead among the MPs are very small. So, the core layer can be very large, i.e. h_c can be very high.

Table 5.2 Calculation of $\boldsymbol{h_c^{min}}$

h	α	β	d	C	U	P	X	Q	d^{h_e}	h_e	h_c^{min}
10	0.5	0.7	3	2.5	26	36	17.02849	227051.8496	38.60809745	3	6
10	0.5	0.07	3	2.5	26	36	17.02849	22705.18496	12.28977159	2	7
24	0.05	0.07	3	0.5	2	9.4	4.614172	21719639162	22869.69254	9	14
24	0.5	0.7	3	2.5	26	36	17.02849	1.08598E+12	84178.77103	10	13
24	0.5	0.0001	2	0.5	0.3	0.5	0.346574	581.4539984	20.66113084	4	19

5.6.3 Ring Placement Minimizing the number of MPs

Because of the nature of the D_{comp}^{DMCU}, there is a large flat area on both sides of h_c^{min}. So, we can find a range for h_c as h_c^{low} and h_c^{high} where the value of the D_{comp}^{DMCU} just a little bit above the $D_{comp}^{DMCU,min}$, Figure 5.14. A lower number of MPs is better because it reduces the possibility of synchronization problems among them. So, we are more interested in h_c^{low}.

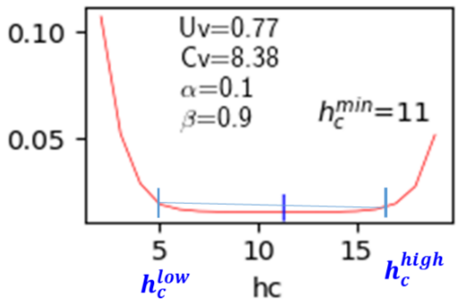


Figure 5.14 The delay graph showing $\boldsymbol{h_c^{min}}$, $\boldsymbol{h_c^{low}}$ and $\boldsymbol{h_c^{high}}$

For the calculation, let us rewrite the equation (5.24) in terms of h_c, ignoring some small terms such as $\frac{1}{U(d-1)^2}, \frac{1}{C}, \frac{1}{C(d-1)}$,

$$D_{comp}^{DMCU} = F\left\{\left(\frac{d^{h-h_c+1}}{U(d-1)^2} - \frac{h}{U(d-1)} + \frac{h_c}{U(d-1)} + \frac{1}{U(d-1)} + \frac{h_c}{U} + \frac{dh}{U} + \frac{\beta d^{h_c+1}}{U(d-1)} - \frac{1}{\mathrm{U(d-1)}}\right) + \frac{d^{h-h_c+1}}{C(d-1)} + \frac{h_c}{C} + \frac{d\alpha h^{h+1}}{C(d-1)^2} - \frac{d\alpha d^{h-h_c+1}}{C(d-1)^2} - \frac{d\alpha h_c}{C(d-1)} + \frac{d\alpha}{C(d-1)}\right\}$$

Or

$$D_{comp}^{DMCU} = F\left\{\left(\frac{d^{h+1}}{U(d-1)^2 d^{h_c}} - \frac{h}{U(d-1)} + \frac{h_c}{U(d-1)} + \frac{1}{U(d-1)} + \frac{h_c}{U} + \frac{dh}{U} + \frac{\beta d^{h_c+1}}{U(d-1)} - \frac{1}{\mathrm{U(d-1)}}\right) + \frac{d^{h+1}}{C(d-1)d^{h_c}} + \frac{h_c}{C} + \frac{d\alpha h^{h+1}}{C(d-1)^2} - \frac{d\alpha d^{h+1}}{C(d-1)^2 d^{h_c}} - \frac{d\alpha h_c}{C(d-1)} + \frac{d\alpha}{C(d-1)}\right\}$$

At h_c^{min}, the slope is zero. We want to move to the left or right of h_c^{min} until the slope is ϵ where the delay is slightly above the minimum. Considering that little bit higher delay, we can choose any value of h_c where,

$$\frac{d(D_i^{max})}{dh_c} < \epsilon, \qquad where\ \epsilon < \frac{\pi}{2}$$

So, we get

$$\frac{d(D_{comp}^{DMCU})}{dh_c} = F\left\{\left(\frac{-\ln(d)\, d^{h+1}}{U(d-1)^2 d^{h_c}} + \frac{1}{U(d-1)} + \frac{1}{U} + \frac{\ln(d)\, \beta d^{h_c+1}}{U(d-1)}\right) + \frac{-\ln(d)\, d^{h+1}}{C(d-1)d^{h_c}} + \frac{1}{C} + \frac{\ln(d) d\alpha d^{h+1}}{C(d-1)^2 d^{h_c}} - \frac{d\alpha}{C(d-1)}\right\} < \epsilon$$

Or,

$$\frac{-\ln(d)\, d^{h+1}}{U(d-1)^2 d^{h_c}} + \frac{\ln(d)\, \beta d^{h_c+1}}{U(d-1)} + \frac{-\ln(d)\, d^{h+1}}{C(d-1)d^{h_c}} + \frac{\ln(d) d\alpha d^{h+1}}{C(d-1)^2 d^{h_c}} < \frac{\epsilon}{F} - \frac{1}{U(d-1)} - \frac{1}{U} - \frac{1}{C} + \frac{d\alpha}{C(d-1)}$$

Or,

$$\frac{-\ln(d)C\,d^{h+1} + \ln(d)C(d-1)\,\beta d d^{2h_c} - \ln(d)U(d-1)\,d^{h+1} + \ln(d)Ud\alpha d^{h+1}}{UC(d-1)^2 d^{h_c}}$$

$$< \frac{\epsilon CU(d-1) - FC - FC(d-1) - FU(d-1) + FUd\alpha}{FCU(d-1)}$$

Or,

$$\frac{-\ln(d)C\,d^{h+1} + \ln(d)C(d-1)\,\beta d d^{2h_c} - \ln(d)U(d-1)\,d^{h+1} + \ln(d)Ud\alpha d^{h+1}}{(d-1)d^{h_c}}$$

$$< \frac{\epsilon CU(d-1) - FC - FC(d-1) - FU(d-1) + FUd\alpha}{F}$$

Or,

$$F\ln(d)C(d-1)\,\beta d d^{2h_c} - (\ln(d)C + \ln(d)U(d-1) - \ln(d)\,Ud\alpha)Fd^{h+1}$$

$$< (\epsilon CU(d-1) - FC - FC(d-1) - FU(d-1) + FUd\alpha)(d-1)d^{h_c}$$

Or,

$$\boldsymbol{Xd^{2h_c} + Pd^{h_c} + Q < 0} \qquad (5.28)$$

Where X = $F\ln(d)C(d-1)\,\beta d$, P = - $(\epsilon CU(d-1) - FC - FC(d-1) - FU(d-1) + FUd\alpha)(d-1)$ and Q =$-(\ln(d)C + \ln(d)U(d-1) - \ln(d)\,Ud\alpha)Fd^{h+1}$

So,

$$d^{h_c} < \frac{-P \pm \sqrt{P^2 - 4XQ}}{2X}$$

Here, Q is a very large negative number, so $\sqrt{P^2 - 4XQ}$ is always greater than P. Because of the exponential nature of the delay function, the value of ϵ is also large ($\frac{de^x}{dx}$ =

e^x, and in the delay equation, the value of d will deviate too much from e). This makes P a large negative number, and X is always positive. So, we can write

$$d^{h_c} < \frac{P \pm \sqrt{P^2 + 4XQ}}{2X}$$

$$\boldsymbol{h_c > \log_d \left| \frac{P - \sqrt{P^2 + 4XQ}}{2X} \right| \approx h_c^{low}} \qquad (5.29)$$

$$\boldsymbol{h_c < \log_d \left| \frac{P + \sqrt{P^2 + 4XQ}}{2X} \right| \approx h_c^{high}} \qquad (5.30)$$

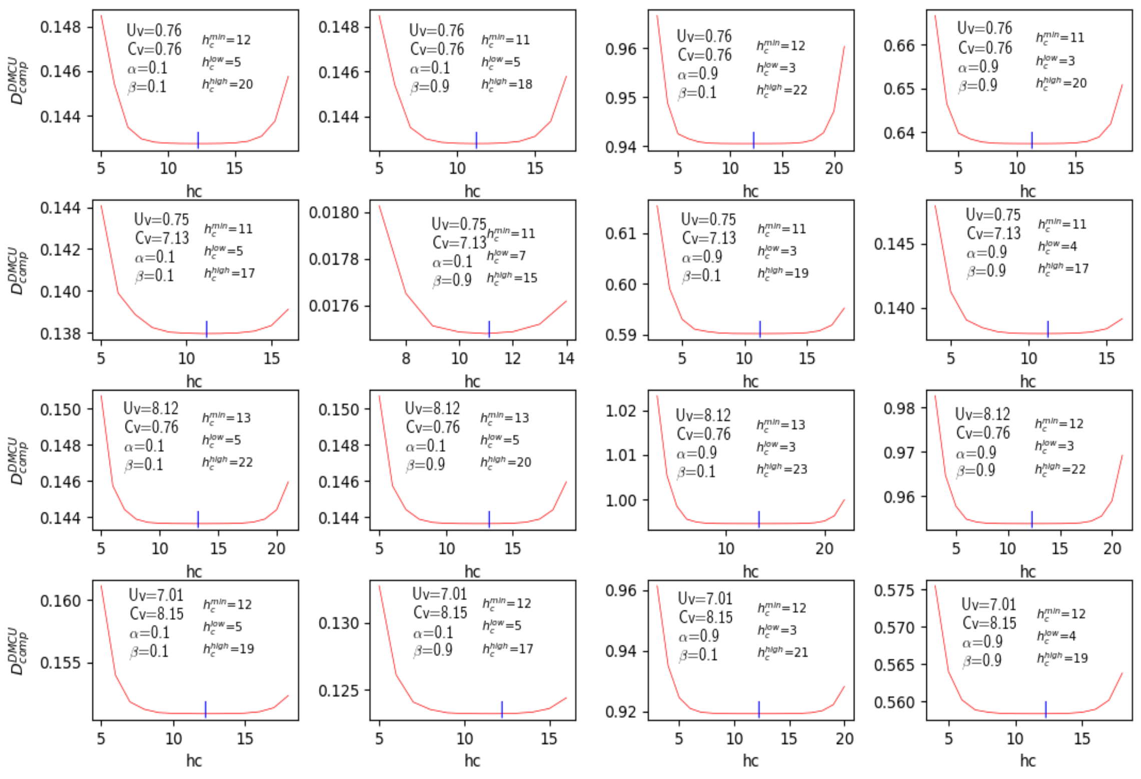


Figure 5.15 $\boldsymbol{h_c^{low}}$ and $\boldsymbol{h_c^{high}}$ for delay between $\boldsymbol{D_{comp}^{DMCU,min}}$ and its 2.5% elevation with different values of $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{U_v}$, and $\boldsymbol{C_v}$

We should choose the size of the core, i.e. h_c near h_c^{low}, because it minimizes delays as well as the number of MPs reducing the possible issues of synchronization among the MPs.

Figure **5.15** shows the plots only region of D_{comp}^{DMCU} where its maximum value only 2.5% above its min value. It also shows the h_c^{low} and h_c^{high}. It means the value D_{comp}^{DMCU} between h_c^{low} and h_c^{high} is less than or equal to 1.025× $D_{comp}^{DMCU,min}$. So, when the delay is only 2.5% above the minimum point, the value h_c^{low} can be as low as 3. So, placing a ring of MPs just 3 to 5 hops away from the MC can significantly reduce the stream composition time. Theoretically, it should minimize 97.5% of delay compared to the system with a single MC processing everything.

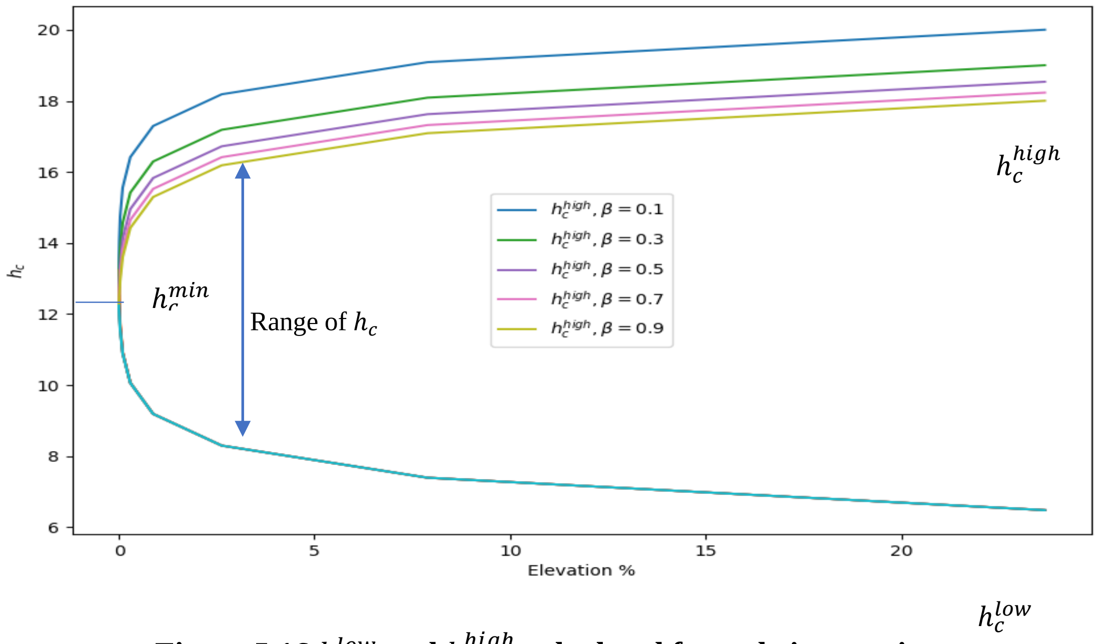


Figure 5.16 h_c^{low} and h_c^{high} calculated from their equations

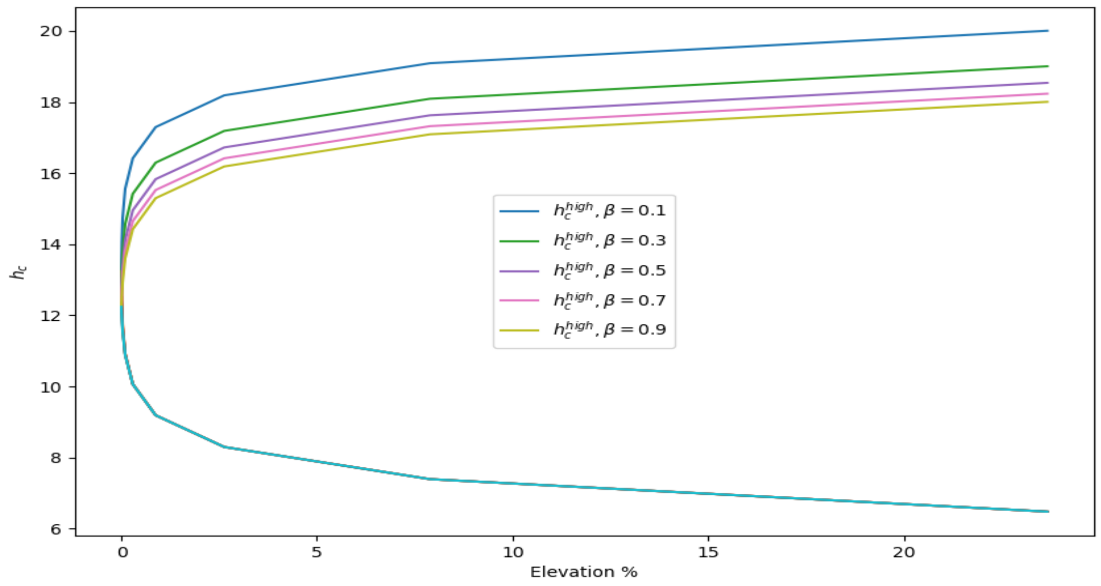


Figure 5.16 shows $\boldsymbol{h_c^{low}}$ and $\boldsymbol{h_c^{high}}$ calculated from their equations for different elevation of $\boldsymbol{D_{comp}^{DMCU}}$ from its minimum value. The values are generated and plotted for different values of $\boldsymbol{\beta}$. From the original equation, we know that $\boldsymbol{\beta}$ affects only second part of the U-shaped graph. Thus, only the $\boldsymbol{h_c^{high}}$ changes for different values of $\boldsymbol{\beta}$. As we see, to keep the composition time within the 2.5% of the minimum value of the branch delay, the $\boldsymbol{h_c}$ must be set within the range 8 and 16. But for the minimum number of MPs, $\boldsymbol{h_c^{low}}$ which is 8 should be picked. Therefore, the ring of MPs should be placed at the 8-hop distance from the MC when the actual height of SCDT *h=24*. This way, the mean delay from all nodes to the MC will be minimized, i.e., the DRDI will be maximized.

5.7 Experimental Results and Discussion

5.7.1 Experimental Setup

For the experiment, the system is run as discussed earlier with up to 50 nodes in the session. The participant nodes incrementally form the topology, as shown in Figure **4.11**. The ZePoP protocol is used to maintain the network and select the MC node. For every network size, the ring is placed at $h_c = 3$, i.e. all nodes that are 3 hop distance away from the MC are made MPs. The dummy streams are sent in 4KB packets. A fixed large waiting time, 200ms, is used at each MPs. For the performance comparison, the total traffic, node hotness and the composition time are recorded.

5.7.2 Comparison on Total Traffic, Hotness and Composition Time

The performance of ring placement technique is compared with all the methods discussed earlier in terms of total traffic, hotness of intermediate nodes and the composition time. Only the Static MC case is not shown in the comparison because the system was crashing with Static MC for transferring large(4KB) packets of streams. Figure **5.17** shows the comparison on total traffic among the all methods including ring placement. In ring placement, there some MPs but the number is less than the number of MPs in Static MC DMP, ZePoP MC-DMP and ZePoP MC-DMPA. Therefore, it reduces the total traffic significantly compared to single MCU based ZePoP MC, but not much compared to other distributed MCU settings. A similar reduction is also observed in terms of hotness of intermediate nodes, Figure **5.18** and the composition time, Figure **5.19**.

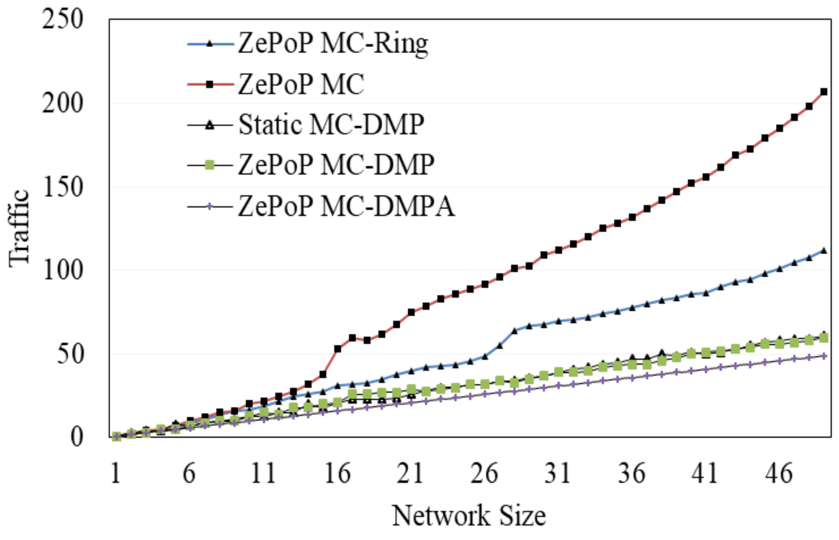


Figure 5.17 Comparison on total traffic

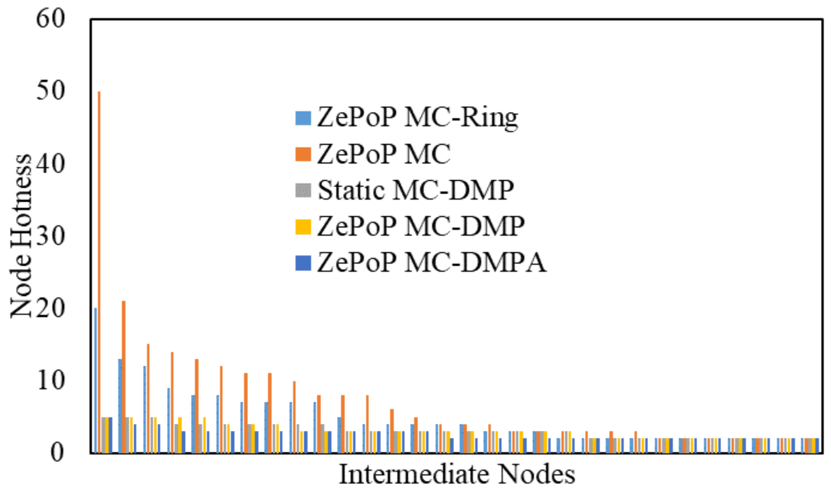


Figure 5.18 Comparison on hotness of intermediate nodes.

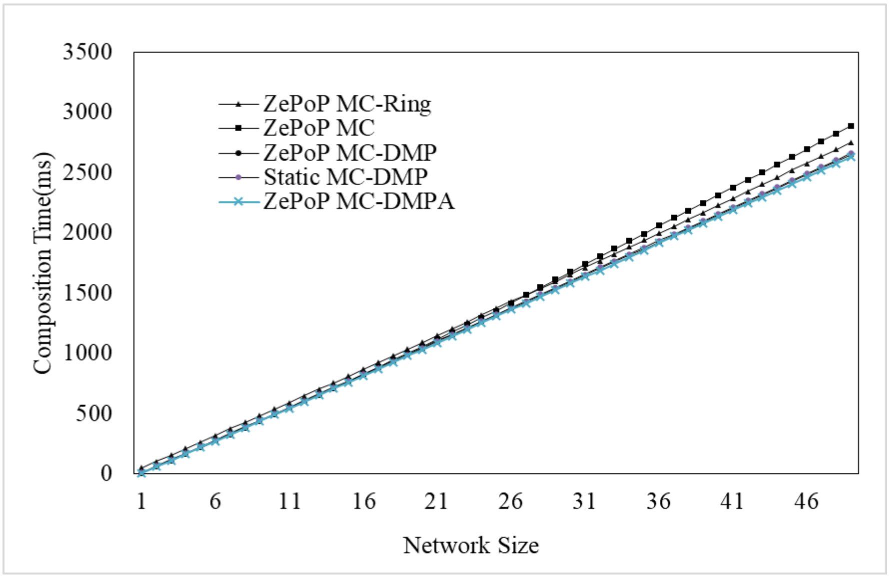


Figure 5.19 Comparison on Composition Time

5.8 Conclusion

In this chapter, a couple of strategies to form distributed MCU are discussed. In the first method, the MPs of the distributed MCU are placed in the SCDT satisfying bandwidth and computational constraints of the participating nodes. The experimental result validates that the distributed MCU can improve the system performance over the centralized MCU by minimizing total traffic, node hotness, composition time, frame loss, i.e., maximizing the DRDI. The wait time management scheme proposed in this chapter helps further to improve the DRDI. However, constraint satisfaction is found to be a complicated and time-consuming task. Also, the optimal delay or DRDI is not guaranteed. So, the second method, called the ring placement technique, is presented. In this method, a ring of MPs is placed on the SCDT from a certain distance from the MC. The distance is estimated from the maximum branch delay in the SCDT when the first method is used. The ring placement calculation aims to minimize that maximum branch delay. The method is validated using both simulation and experiment. The results suggest a range of values of the ring distance. Any of these values are supposed to give the composition time very close to the optimum value. However, to keep the number of MPs small, the lowest value of the range should be used. The adaptive wait time management scheme assigns the timers at the MPs based on the path delays in the network. So, some slow nodes in the network can significantly increase the waiting time. Also, as the number of nodes increases, the adaptive timer will increase. The waiting time should not be too high because it will make the composition time or frame interval very high. Therefore, it will decrease the PCE. The adaptive timer management scheme will work better when the network delays are relatively small compared to the desired composite frame interval. But for a CMTS with an overcrowded situation, the system must work based on a fixed and reasonably short waiting time provided at the MC to maintain the high PCE or the frame rate when required. In the next chapter, an optimal timer scheme is presented that uses the given waiting time for assigning optimal timer at the MPs. The optimal timer scheme is expected to yield high PCE or profit, ensuring the desired frame rate or composition time.

CHAPTER 6

Differentiated Role-based CMTS

6.1 Introduction

This chapter addresses the final phase(Phase3), relaxing the fourth(final) assumption, i.e., all the four assumptions and presents the final differentiated role-based design of the CMTS. The ZePoP relaxes the first assumption by using the link delays for creating the optimal SCDT. This phase extends the ZePoP protocol and presents the final version called $ZePoP$-ϵ protocol that removes assumption 4 by including stream scores of the participant nodes in the objective function. The stream score of a participant is calculated based on the profile weight and the demand score of the participant and used to prioritize the stream of the node. The protocol places the MC in such a way that only the streams with high priority can travel and reach the MC within a bounded waiting time. It helps to keep the low priority participant nodes far from the MC so that their streams can be restricted to travel through the network and consume the network resources. This is how the protocol addresses the overcrowding challenge of CMTS. After creating the SCDT by maximizing a role-based profit function, the MPs are placed in the SCDT using the first technique presented in CHAPTER 5, i.e., satisfying constraints. Thus, the second assumption is also removed. The ring placement technique can also be used to remove assumption 3. The adaptive waiting time management discussed in the previous chapter uses a bottom-up approach to assign the waiting time at the MPs. The timer of an MP is calculated based on the maximum path delay in its subtree. So, some slow nodes in the network can significantly increase the waiting time at the MPs. Also, as the number of nodes increases, the adaptive timer will increase. The waiting time should not be too high because it will make the composition time or frame interval very high. Therefore, it will decrease the PCE. The adaptive timer management scheme will work better when the network delays are relatively small compared to the desired composite frame interval. But for a CMTS with an overcrowded situation, the system must work based on a fixed and reasonably small waiting time provided at the MC to maintain the high PCE or the frame rate when required. In this chapter, an optimal timer scheme is presented that uses the given waiting time for assigning optimal waiting time at the MPs. The optimal timer scheme is expected to yield high PCE or profit/DRDI, ensuring the desired frame rate or composition time.

6.2 ZePoP-ϵ Protocol

The $ZePoP$-ϵ is the extended protocol of ZePoP, so it inherits all the supporting features of ZePoP such as node joining, node departure, election initiation, etc. The message scheme is also the same with some modification in the structure of ELECTION, JOIN and INFORM message. The ELECTION and JOIN messages have an additional field to carry the **stream score(p)** of the message's source node. The centrality(C) field of the INFORM message is renamed to Stream Score(P). The objective or profit function is modified to include stream scores of participants and the election algorithm is also modified to adapt with the modified objective function. An extra step is added for reducing the number of MC candidates. The protocol creates the optimal SCDT maximizing the total profit at the MC node.

6.2.1 Estimation of Profile Weight

The profile weight w_i of a participant, *i* can be calculated based on the following two role-based metrics,

a. Global Role (G_i): This is a global score, and it can be calculated based on the offline records as well as online profiles such as google scholar citations, number of followers in Facebook profiles or pages, subscribers or followers on YouTube channels, etc.
b. Session Role (S_i): This is a local score that is assigned to each participant of a session. The conference organizer can assign scores to the different roles of a multi-session conference. The common roles are keynote speaker, ordinary participant, session chair, presenter, committee member, conference chair, etc. A participant receives a score based on his or her role in a session. So, a participant can receive a different score in different sessions.

Knowing the values of G_i and the S_i, the profile weight can be calculated as $w_i = f(G_i, S_i)$. However, determining the function *f* and the estimation of the actual value of G_i and S_i is extremely complicated and needs further research. In this dissertation, the discussion is kept limited here, and randomly generated values of profile weights are used in the simulations and emulations.

6.2.2 Demand Score

The demand score v_i of a participant, *i* is the number of requests the node *i* receives from other participants to send its stream. A peer or participant can send a stream request to another participant directly or via the MC node. In this dissertation, the simulated value of v_i is used.

6.2.3 Formulation of the Problem and the Objective Function

Suppose an undirected graph *G = (V, E)* represents a topology of a telepresence session on peer to peer network where *V* is the set of peers and *E* is the set of edges among peers. Each participant has a stream score calculated from the demand score and the profile weight. We must find an MC peer $m \in V$ so that during stream composition, the total score or profit of all streams at MC is maximized.

Suppose there are *n* participants in G where $n \geq 2$. Each participant *i* maintains w_i which is the profile weight and a demand score v_i. The stream score of node *i* is calculated as $p_i = w_i v_i$. During the telepresence session, each participant peer originates a raw stream of rate *s*, forwards streams of other peers, and can work as an MC when required. The raw streams of participants are collected by the MC peer and then mixed into one composite video stream also of rate *s*, which is eventually delivered to all participant peers. For each stream composition, the MC waits for a certain period W, and the composition includes only the streams that reach before the waiting time expires. So, the total profit accumulated by the MC is defined as the equation 3.2), and it is shown below,

$$\boldsymbol{P_M(W) = \int_0^W \sum_{s \epsilon V} w_s v_s f_{s,M}(t) dt} \qquad (6.1)$$

Where $f_{s,M}(t)$ is the probability distribution function of path delays from node s to the MC node M. However, if the W is very high, then the profit accumulation will be maximum and equal at each node. So, the closeness centrality should come into play to pick the better node as MC. So, the modified profit or objective function is,

$$P'_M(W) = C_M \int_0^W \sum_{s \epsilon V} w_s v_s f_{s,M}(t) dt$$

$$\boldsymbol{= \frac{n-1}{\sum_{s \epsilon V, s \neq M} D_{s.M}} \int_0^W \sum_{s \epsilon V} w_s v_s f_{s,M}(t) dt} \qquad (6.2)$$

The MC election algorithm must select a participant peer $M \in V$ as the MC so that the profit $P'_M(W)$ is maximized.

6.2.4 Proposed Solution

For simplicity, let us rewrite the equation (6.2) in the discrete domain as shown below,

$$\boldsymbol{P_M^W = \frac{n-1}{\sum_{s \epsilon V, s \neq M} D_{s.M}} \times \sum_{s \epsilon V} w_s v_s H(W - D_s)} \qquad (6.3)$$

Where the D_s is the path delay from node s to the MC. The function $H(*)$ is Heaviside step function defined below for discrete variable K,

$$\boldsymbol{H(K)} = \begin{cases} \mathbf{0}, & \boldsymbol{if\ K < 0} \\ \mathbf{1}, & \boldsymbol{if\ K \geq 0} \end{cases} \tag{6.4}$$

So, if $D_s > W$ then the stream score of node *s* will not be accumulated at the node M. For the MC election in the dynamic P2P network, we can use all the supporting algorithms from ZePoP. So only the MC election algorithm is discussed in this section. Suppose a node *k* accumulates the stream scores from the nodes in R_k. Then, in the election algorithm, the following simple operations can be performed,

a. The MC detects the MC election situation and broadcasts the ELECTION message to its neighbors.

b. When a node k receives an ELECTION message of node *s*, it can collect the stream score contained in the message into R_k. But it makes sure that the score is collected only once from each source *s*. The ELECTION message contains stream score p_s as well as the shortest path delay from the node s to k, D_s. Then, it can sum up the profit as follows,

$$\boldsymbol{R_k = R_k + p_s H(W - D_s)} \tag{6.5}$$

c. Then it forwards that ELECTION message to the neighbors and broadcasts its own ELECTION message. Any better ELECTION message (in terms of path delay) from the same source is always forwarded.

d. After receiving the stream score from each node in the system, the node *k* can calculate its profit earning as defined in (6.3), i.e.,

$$\boldsymbol{P_k^W} = \frac{(\boldsymbol{n}-\mathbf{1}) \times \boldsymbol{R_k}}{\sum_{s\epsilon V, s\neq k} \boldsymbol{D_{s.k}}} \qquad (6.6)$$

e. At this point, each node k, share its profit P_k^W with its all neighbor in G.

f. A node *k* identifies it as an MC candidate iff $P_k^W \geq P_j^W$ for all its neighbors *j* in the topology *G*. If for two neighbors x and y, $P_x^W = P_y^W$ then the lowest ID, i.e., $\min(x, y)$, is the MC candidate.

g. Each candidate node declares its candidacy by broadcasting the INFORM message. The INFORM message contains the calculated profit of the candidate.

h. After receiving the INFORM messages, each node picks the candidate with the highest profit as the new MC.

As the new MC is elected, the SCDT is formed using the same process, as explained in ZePoP.

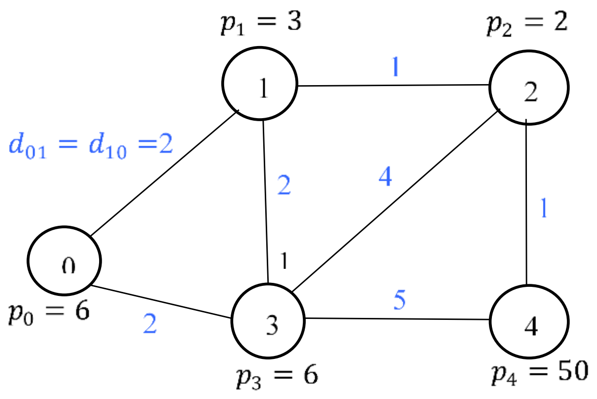


Figure 6.1 A sample topology for an illustrative example

6.2.5 An Illustrative Example

For understanding the algorithm, let us discuss it using an example. Suppose we have a P2P topology of 5 nodes, as shown in Figure 6.1. The blue numbers on the links are the link latency and p_i is the stream score of node i. Now, we can check which node can be the MC for different values of waiting time W. If W=1 then,

$$R_0 = p_0 H(1-0) + p_1 H(1-2) + p_2 H(1-3) + p_3 H(1-2) + p_4 H(1-4)$$

$$\text{Or } R_0 = 6$$

So, no one can reach node 0 within the waiting time W=1. So, the profit collection at node 0 is,

$$P_0^1 = \frac{4 \times 6}{11} = 2.18$$

Table 6.1 Calculations for MC election Example

Nodes	**W=0**	**W=1**	**W=2**	**W=3**	**W=4**
0	$P_0^0 = 2.18$	$P_0^1 = 2.18$	$P_0^2 = 5.45$	$P_0^3 = 6.18$	$P_0^4 = 24.36$
1	$P_1^0 = 1.71$	$P_1^1 = 2.85$	$\boldsymbol{P_1^2 = 38.28}$	$\boldsymbol{P_1^3 = 38.28}$	$\boldsymbol{P_1^4 = 38.28}$
2	$P_2^0 = 1.0$	$\boldsymbol{P_2^1 = 27.5}$	$P_2^2 = 27.5$	$P_2^3 = 33.5$	$P_2^4 = 33.5$
3	$P_3^0 = 2.18$	$P_3^1 = 2.18$	$P_3^2 = 5.45$	$P_3^3 = 6.18$	$P_3^4 = 24.36$
4	$P_4^0 = 18.18$	$P_4^1 = 18.9$	$P_0^2 = 20$	$P_4^3 = 20$	$P_4^4 = 24.36$

The similar calculations for each node with different values of W are shown in Table 6.1. For each value of W, the node with the highest profit value would be elected as the MC node. The profits of such nodes are marked red in the table. It can be observed that, as the waiting time W increases, the profit value increases. The MC also moves towards the node with the highest closeness centrality (node1). For $W \geq 4$, each node will able to collect highest possible profit, but MC will be elected based on the closeness centrality. So, for a CMTS session with a very high waiting time, compared to the shortest path delays of participants, the MC will be decided based on the closeness centrality. But for a CMTS session with a very large number of participants and W is small compared to the path delays, the MC will move towards the region where many participants have high stream score so that the overall profit can be maximized. Thus, the overcrowding issue is addressed by avoiding low profile participants. The MC node can send a notification to those low-profile nodes so that they do not generate streams. Thus, the system can utilize its resources only for high profile participants.

6.2.6 Simulation Results and Discussion

For observing the performance of the proposed protocol, it is simulated based on the same topology, as shown in Figure **4.11**. The topology has 50 nodes, i.e., n = 50. For each node s, the profile weight w_s and the demand score v_s are randomly generated from range [0, 50] and [0, n) respectively. Then the stream scores of the node are calculated as $p_s = w_s v_s$. The link delays are also generated randomly from the range [0.01, 5]. For comparative analysis, node 0 is considered as the static MC since it is the initial node. In the simulation, as a new node joins, the network size as well as shape change. The new nodes are connected according to the topology under consideration. The new election situation can be detected based on the MC candidacy violation method. In this work, it is considered that the election is performed after each node arrives. So, each election aims to move the MC to form the new SCDT maximizing the DRDI or the accumulated profit.

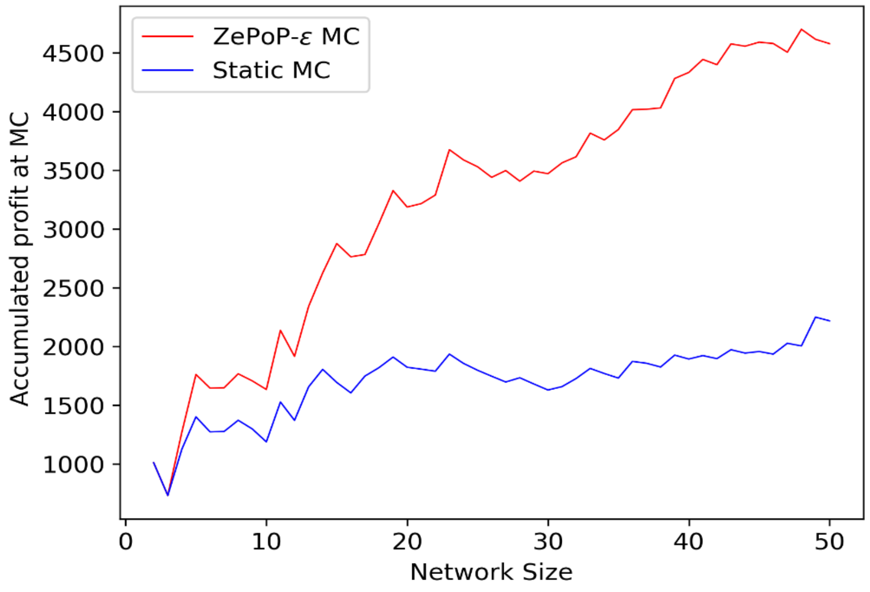


Figure 6.2 Profit at Static and $\boldsymbol{ZePoP}$-$\boldsymbol{\epsilon}$ MC as the network grows

Figure **6.2** compares the accumulated profit at $ZePoP$-ϵ MC and the Static MC against the network size. As the network grows, i.e., the new nodes join, re-elections are performed to move MC dynamically. During the elections, the MCs in both cases accumulate the profits based on the given waiting time. For this experiment, the waiting time is assigned to 85% of the diameter (in terms of delay). The diagram shows that the $ZePoP$-ϵ MC can accumulate significantly higher profit with the same waiting time than the Static MC. Thus, the $ZePoP$-ϵ MC can always improve the system performance and satisfy the desired quality of service defined within a bounded waiting time.

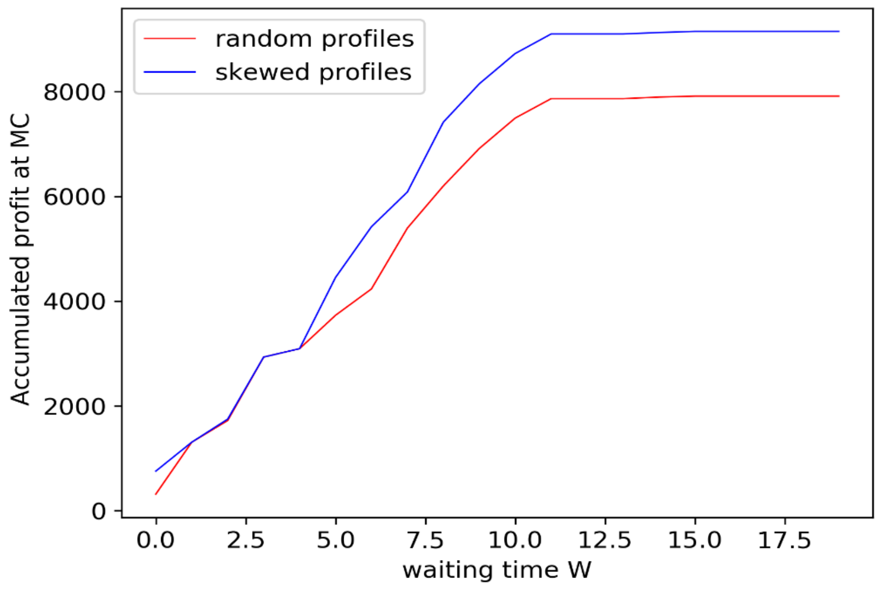


Figure 6.3 Accumulated profit at the $\boldsymbol{ZePoP}$-$\boldsymbol{\epsilon}$ MC against the waiting time

Initially, the stream scores are assigned randomly for all the nodes. For observing the impact of skewed profiles (some participants with very high stream scores), node 0 is set with a very high stream score. Then, for different values of the waiting time W, the MC movements, and the profit accumulations are observed. Figure **6.3** shows that the $ZePoP$-ϵ MC node always maximizes the profit accumulation by moving the MC as necessary. In other words, as the waiting time increases, the profit accumulation also increases for both random and skewed cases because the MC is moved by the $ZePoP$-ϵ election algorithm. However, after a specific value of the waiting time, the profit accumulation becomes saturated because, at this point, each node in the network has the same accumulated stream scores, but only one node is best in terms of closeness centrality and the system already found that node.

Table 6.2 The movement of MC with random profiles(RP) and skewed profiles(SP)

	MC movement with Waiting Time W ----->																			
W	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
RP-MC	7	26	19	24	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
SP-MC	0	4	6	9	13	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16

Table 6.2 shows the movement of MC as the waiting time W increases. As we can see, the small values of W keep the MC near the heavily scored participant 0 (skewed case) so that the profit is always maximized. But as the W grows, the MC moves towards the center node having the highest closeness centrality. Once the MC reaches the center node, the total profit starts becoming saturated, and the MC never moves, even the waiting time keeps increasing.

6.3 Placement of MPs and Assigning Optimal Timer

The MC placement algorithms in $ZePoP\text{-}\epsilon$ create the optimal SCDT maximizing the total profit or DRDI at the MC. For CMTS, we need to place the MPs, as discussed in the previous chapter. We have observed that the MPs can significantly reduce the shortest path delays of the nodes to the MC. It will help to deliver more participants to the MC. i.e., further, maximize the profit. The waiting time management discussed in the previous chapter uses a bottom-up approach to assign the waiting time at the MPs. So, the system might become very slow if multiple nodes stay slow for a long time. Also, it does not directly use the given waiting time for wait time management. In this section, an optimal timer management scheme is discussed that uses the given waiting time at MC to assign the waiting time at the MPs and maximize the DRDI. The DRDI used for the optimality is formulated in the next sections. A version of this optimal timer scheme is discussed in [83]. The optimal scheme is shown as best compared to other heuristics-based timer schemes.

6.3.1 Problem Formulation

Suppose in a CMTS; there are n participants, $n \geq 2$. The network topology is represented by graph G = (V, E). Each participant *i* maintains w_i which is an individual profile weight and v_i which is the demand score. So, the stream from node *i* is associated with a score $p_i = w_i v_i$.

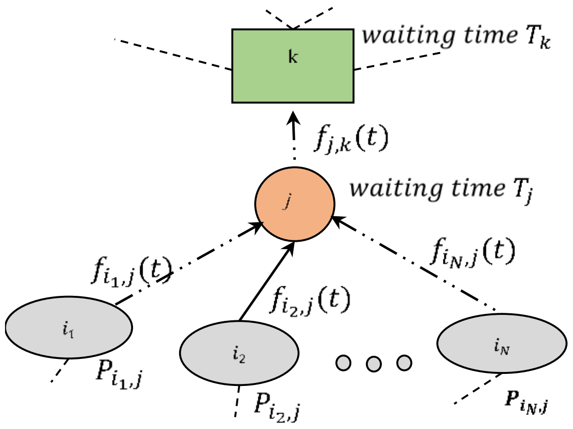


Figure 6.4 A part of SCDT

The SCDT formed on the G is optimal, which ensures a maximum profit or DRDI at the dynamically elected MC. There are one or more MPs placed in the SCDT to form the distributed MCU. Now, given the waiting time W at the MC, we have to assign optimal waiting time at the MPs so that the profit or DRDI is further maximized.

Let us consider a part of the SCDT, as shown in Figure 6.4. The nodes shown in the figure are the MPs or leaf nodes. There might have one or more non-MP nodes along a path from an MP to another show in the dotted line. Node *k* is an MP, and it is the immediate ancestor of another MP node *j*. The node *j* is the immediate descendant of the node *k*. The node *j* has *N* direct descendant nodes $i = \{i_1, i_2 \cdots i_N\}$. Let $P_{i_x,j}$, x = {1,2, · · · N} be the profit or DRDI carried with the streams from node i_x to node j. Let $f_{i_x,j}(t)$ be the probability distribution function of path delay from nodes i_x to parent node *j*. $f_{j,k}(t)$ is the probability distribution of path delays from node *j* to node *k*. Then, one step of optimal timer assignment can be defined as, "*Given the waiting time T_k at node k we have to assign the waiting time T_j at node j so that the total profit from node j to node k is maximized*". The optimal timer assignment at MPs will be multiple such steps to be performed sequentially from top to bottom.

6.3.2 Profit Function

The profit or DRDI function can be defined as the product of profit accumulation and the probability of successful delivery of profit (from the streams) to the next ancestor node. Let $\int_0^{T_j} C(t)dt$ be the total profit accumulation at node *j* in time T_j and $\int_0^{T_k - T_j} S(t)dt$ probability of successfully reaching the ancestor *k* in time $(T_k - T_j)$ with the accumulated profit. The node *j* forwards the accumulated profit to the node *k* only when the profits arrive from all its immediate descendants or the waiting time T_j expires. The accumulated profit also includes the stream score of node *j*. So, the profit or DRDI carried from *j* to *k* is,

$$P_{j,k} = \int_0^{T_j} C(t)dt \int_0^{T_k - T_j} S(t)dt$$

$$= \left(\int_{\mathbf{0}}^{\boldsymbol{T_j}} \sum_{\boldsymbol{s \epsilon A}} \boldsymbol{P_{s,j}\, f_{s,j}(t)dt} \right) \int_{\mathbf{0}}^{\boldsymbol{T_k - T_j}} \boldsymbol{f_{j,k}(t)dt} \qquad (6.7)$$

Where $A = \{j\} \cup \{i_1, i_2, \dots i_N\}$.

Figure **6.5** shows the formulation of the profit function. The area under the curves $f_{i_1,j}$ and $f_{i_2,j}$ are the probabilities of reaching the descendants i_1 and i_2 at node j with their profit, (a) and (b). As the waiting time T_j increases, the areas under these curves increase. It corresponds to the profit increment at the node *j*. In other words, as the waiting time T_j increases the profit accumulation $\int_0^{T_j} C(t)dt$ at the node j increases. However, the diagram (c) and (d) show that as the waiting time T_j increases the area under the curve $f_{j,k}$ decreases because T_k is fixed. This area is the probability of reaching node *j* to *k* with the accumulated profit. So, with the increment of T_j, the probability of reaching the node *j* to *k* will decrease, i.e., the profit accumulation at node *k* will decrease. The diagram (e) shows that the accumulated profit C(t) initially increases with T_j but eventually become saturated. Also, the probability of successful delivery S(t) decreases with T_j. So, there must be a value of T_j, say T_{opt}, where the product $\int C(t) \int S(t)$ will be maximum (f).

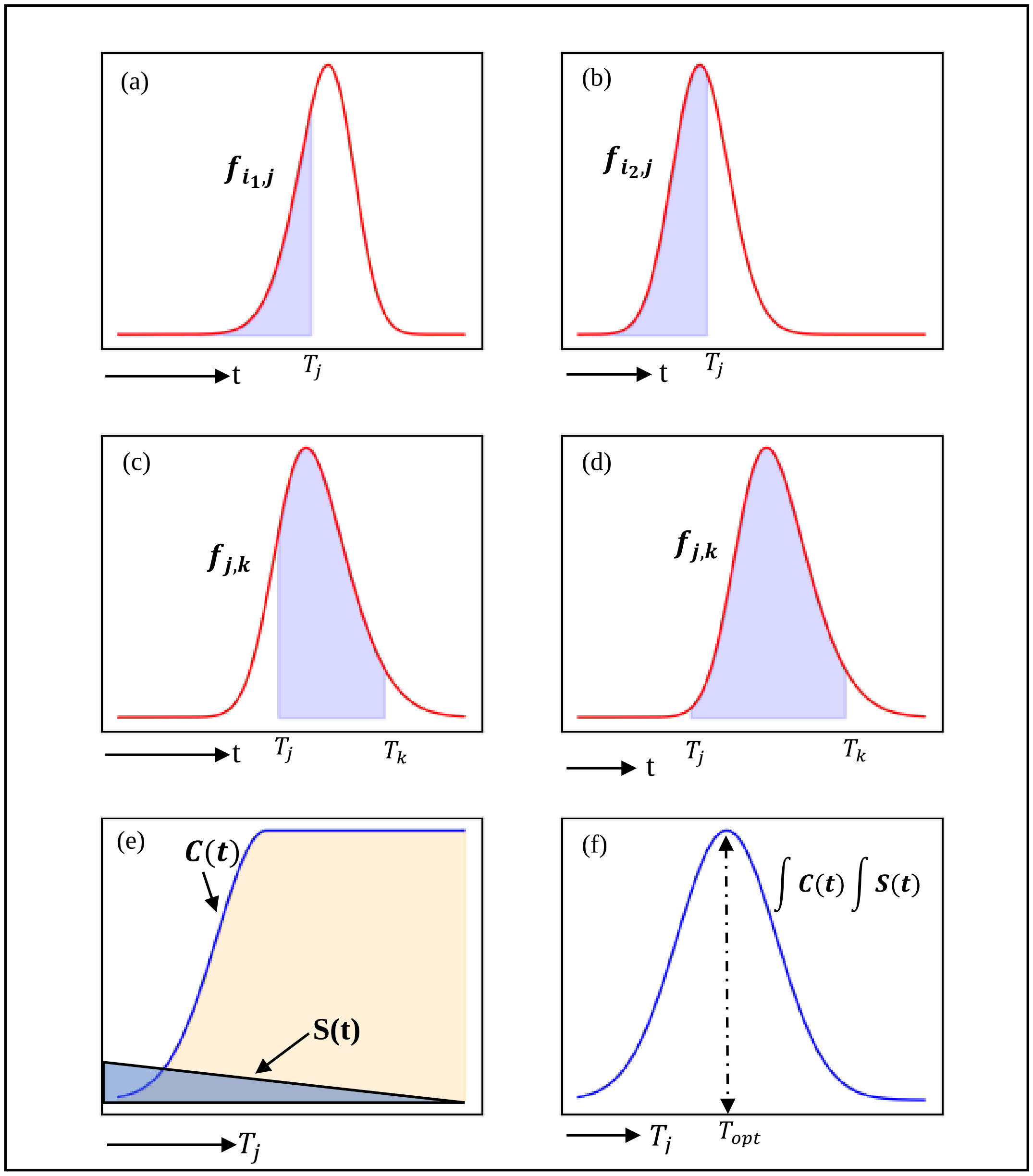


Figure 6.5 Formulation of Profit Function

So, given T_k at the node *k*, we have to find $T_j = T_{opt}$ so that $P_{j,k}$ is maximized. In general, given the waiting time *W* for the MC node *M*, we have to assign optimal waiting time at all MPs so that the total profit or DRDI at MC is maximized.

6.3.3 Proposed Solution

Suppose the optimum timer T_{opt} at the node *j* maximizes the profit from *j* to *k*, and it is represented by $P_{j,k}^{max}$. For calculating the T_{opt} and $P_{j,k}^{max}$, let us assume,

a) The delay distributions of immediate ancestor and the descendants of node *j* are of Normal Distribution, e.g. $D_{i_1,j} \approx N(\mu_{i_1,j}, \sigma_{i_1,j}^2)$ and $D_{j,k} \approx N(\mu_{j,k}, \sigma_{j,k})$.
b) The distributions of path delays are independent of each other.
c) Node *j* always forwards the profit to node *k*, either after the waiting time expires or it receives profits from all the immediate descendants.

It is known that the summation of normal random variables follows the normal distribution, i.e.,

$$\sum \boldsymbol{P_i X_i} = \boldsymbol{N}(\sum \boldsymbol{P_i \mu_i}, \sum (\boldsymbol{P_i \sigma_i})^2) \qquad (6.8)$$

where, $\boldsymbol{X_i} \approx \boldsymbol{N}(\boldsymbol{\mu_i}, \boldsymbol{\sigma_i})$. So, the equation (6.7) can be written as.

$$\boldsymbol{P_{j,k}} = \left(\int_{0}^{T_j} \boldsymbol{f_j(t)dt} \right) \int_{0}^{T_k - T_j} \boldsymbol{f_{j,k}(t)dt} \qquad (6.9)$$

Where the path delay D_j is the path delay from direct descendants to node *j* and,

$$D_j \approx N(\mu_j, \sigma_j^2), \qquad \mu_j = \sum_{s \epsilon A} P_{s,j} \mu_{s,j} \ and\ \sigma_j^2 = \sum_{s \epsilon A} (P_{s,j} \sigma_{s,j})^2$$

Though the node k may have multiple descendants like node j, here we discuss only the profit delivery from node j to node k. So, let us rewrite $f_{j,k}(t)$ as $f_k(t)$ where $u_k = \mu_{j,k}$ and $\sigma_k^2 = \sigma_{j,k}^2$. So, the equation 6.9 can be written as follows,

$$\boldsymbol{P_{j,k}} = \left(\int_{\mathbf{0}}^{\boldsymbol{T_j}} \boldsymbol{f_j(t)dt}\right)\int_{\mathbf{0}}^{\boldsymbol{T_k-T_j}} \boldsymbol{f_k(t)dt} \qquad (6.10)$$

As $D_j > 0$ and $D_k > 0$, then the means $\mu_j > 0$ and $\mu_k > 0$. Therefore, $\int_{-\infty}^{0} f_j(t)dt$ and $\int_{-\infty}^{0} f_k(t)dt$ are negligible. Then the equation (6.10) can be written as,

$$\boldsymbol{P_{j,k}} = \left(\int_{-\infty}^{\boldsymbol{T_j}} \boldsymbol{f_j(t)dt}\right)\int_{-\infty}^{\boldsymbol{T_k-T_j}} \boldsymbol{f_k(t)dt} \qquad (6.11)$$

$$= \int_{-\infty}^{T_j} \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{\left(t-\mu_j\right)^2}{2\sigma_j^2}} dt \int_{-\infty}^{T_k-T_j} \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{(t-\mu_k)^2}{2\sigma_k^2}} dt$$

So,

$$\boldsymbol{P_{j,k}} = \frac{\mathbf{1}}{\mathbf{2}}\left[\mathbf{1} + \boldsymbol{erf}\left(\frac{\boldsymbol{T_j - \mu_j}}{\boldsymbol{\sigma_j}\sqrt{\mathbf{2}}}\right)\right] \times \frac{\mathbf{1}}{\mathbf{2}}\left[\mathbf{1} + \boldsymbol{erf}\left(\frac{\boldsymbol{T_k - T_j - \mu_k}}{\boldsymbol{\sigma_k}\sqrt{\mathbf{2}}}\right)\right] \qquad (6.12)$$

As the cumulative distribution function for normal distribution function is $\Phi(x) = \frac{1+\mathrm{erf}(\frac{x}{\sqrt{2}})}{2}$, the equation (6.12) can be represented as,

$$\boldsymbol{P_{j,k}} = \boldsymbol{\Phi}\left(\frac{\boldsymbol{T_j - \mu_j}}{\boldsymbol{\sigma_j}}\right) \times \boldsymbol{\Phi}\left(\frac{\boldsymbol{T_k - T_j - \mu_k}}{\boldsymbol{\sigma_k}}\right) \qquad (6.13)$$

```
CalculateT ($T_k, u_j, \mu_k, \sigma_j, \sigma_k, P_{max}, T_{opt}$)
1. Initialize $T_{opt} \leftarrow 0$ and $P_{max} \leftarrow 0$
2. for $T_j = 0$ to $T_k$ step $\delta$:
3.         calculate $P = \Phi\left(\frac{T_j-\mu_j}{\sigma_j}\right) \times \Phi\left(\frac{T_k-T_j-\mu_k}{\sigma_k}\right)$
4.         if $P_{max} < P$ then:
5.             $P_{max} \leftarrow P$
6.             $T_{opt} \leftarrow T$
```

Figure 6.6 Algorithm for calculating $\boldsymbol{T_{opt}}$ with maximum profit

Because of the nature of the $erf(*)$ function, the profit function $P_{j,k}$ is always greater than or equals to 0 and the continuous function of T_j. As $T_j \rightarrow \pm\infty$, it moves towards zero. Therefore, a maximum profit $P_{j,k}^{max}$ corresponding to $T_j = T_{opt}$ must exist where $\frac{d\left(P_{j,k}^{max}\right)}{dT_j} = 0$. This derivation can be used to find the T_{opt}. There several other methods that can be used to find both the $P_{j,k}^{max}$ and T_{opt}. A simple iterative method can also be used, as shown in Figure **6**.**6**. Any small step value, i.e., $\delta = 0.001$ will allow to get the actual pick of the profit function.

6.3.4 An Illustrative example

To observe the effect of different parameters in the profit function, T_{opt} and corresponding P_{max} are calculated with their different combination of values.

Table 6.3 Simulation of $\boldsymbol{T_{opt}}$ and $\boldsymbol{P_{max}}$, for $\boldsymbol{T_k = 10}$

Case No	$\boldsymbol{\mu_j}$	$\boldsymbol{\mu_k}$	$\boldsymbol{\sigma_j}$	$\boldsymbol{\sigma_k}$	$\boldsymbol{P_{max}}$	$\boldsymbol{T_{opt}}$
1	1	1	1	1	0.999937	5.00002
2	1	1	1	3	0.961097	3.38
3	1	1	3	1	0.961097	6.62006
4	1	1	3	3	0.825897	5.00002
5	1	4	1	1	0.987619	3.5
6	**1**	**4**	**1**	**3**	**0.826224**	**2.77**
7	1	4	3	1	0.826224	4.23
8	1	4	3	3	0.63628	3.5
9	4	1	1	1	0.987619	6.50005
10	4	1	1	3	0.826224	5.77004
11	**4**	**1**	**3**	**1**	**0.826224**	**7.23007**
12	4	1	3	3	0.63628	6.50005
13	4	4	1	1	0.707861	5.00002
14	4	4	1	3	0.535543	5.22002
15	4	4	3	1	0.535543	4.78001
16	4	4	3	3	0.397604	5.00002
17	**8**	**1**	**1**	**1**	**0.47812**	**8.5001**
18	**1**	**8**	**1**	**1**	**0.47812**	**1.5**
19	**26**	**6**	**2**	**1**	**3.99581e-24**	**9.26012**
20	**27**	**6**	**2**	**1**	**0**	**0**

Table 6.3 shows examples of such calculations with $T_k = 10$. For the first 16 calculations, μ_j and μ_k are taken from {1, 4} and σ_j and σ_k are from {1, 3}. In next two cases, μ_j and μ_k are set with big difference between them and very closed to the T_k. It can be observed that, given the fixed value of waiting time T_k, the profit accumulation (P_{max}) from node j to k decreases with increasing values of means and the standard deviations. The high values of σ_k and μ_k compared to σ_j and μ_j lower the T_{opt} for node j and the reverse combination allows it to set a high waiting time at node j (cases 6, 11,

17,18). In the last two cases, the μ_j is set to very high compared to μ_k and T_k. It shows that the model tries hard to assign an optimal timer at the node j, even very close to T_k and very little profit score, but can assign zero at some point($u_j = 27$), with no hope to deliver any profit to node *k*.

6.4 Experimental Result and Discussion

For testing the optimal timer scheme, the CMTS is emulated on the local cluster nodes. The MC can be elected using any one of the three algorithms presented in this dissertation. But for the convenience of comparison of different waiting time management schemes, the ZePoP is used. After electing the MC, the MPs are placed in the SCDT using the constraints satisfaction scheme discussed in 5.3.3. Then the system is run for three different waiting time management schemes (i) Fixed Timer Scheme(FTS) at all the MPs, including the MC (ii) Adaptive Timer Scheme (ATS) and (iii) Optimal Timer Scheme (OTS). These three schemes are compared on different metrics, including composition time and the profit accumulation at the MC node.

6.4.1 Necessary Calculations ($\boldsymbol{\mu_j, \mu_k, \sigma_j, \sigma_k}$)

For FTS, the system settings and the necessary calculations are very simple—only a fixed waiting time W is assigned at each MPs, including MC. For ATS, the waiting time W is calculated as explained in 5.4. The nodal delay and the path delays calculated in (5.7) and 5.9) respectively are also used in OTS for calculating means and standard deviations. A path delay is calculated as the sum of the nodal delays and the link delays along the path.

The nodal delays are calculated, as mentioned in the (5.7). The link delay is approximated when each node joins the CMTS session. When a node connects to a node in the overlay network, they exchange several data packets proportional to the maximum number of connections they can make in the overlay. The time taken for the exchange is considered as the link delay. In the OTS, to deliver maximum profit from node j to k, the waiting time T_j must capture all the descendants in the subtree rooted at *j*. Therefore, the mean u_j must be the average of path delays from all the descendants to the node *j*. Because of the different level of nodes in the subtree, this mean value can be short of capturing the profit of all nodes, as T_j is mostly proportional to the μ_j (as we have seen in Table 6.3). So, the μ_j is calculated as the average of maximum branch delay values recorded over time at node *j*. Note that, because of the variable nodal delays along the path, the maximum branch delay also varies.

For calculating the path delays, the stream packets traveling towards the MC carry the summation of nodal delays. The node *j* calculates the path delay as the summation of the nodal delays and the link delays, which are already known to node *j* during the election. Suppose for a stream composition *t*; the node *j* records the maximum path delay $D_j^{t,max}$ among all path delays D_j^t of the descendants. After each K compositions, the node *j* calculates μ_j as follows,

$$\boldsymbol{\mu_j = \frac{1}{K}\sum_{t=1}^{K} D_j^{t,max}} \qquad (6.14)$$

Then the μ_j and all the recorded delays $D_j^{t,max}$ can be used to calculate the σ_j.

The link delays are assumed fixed for calculating the path delays. So, for estimating u_k, only the nodal delays along the path from j to k are variable. The composite stream packets carry the summation of these nodal delays when they travel from node k towards the node j. The composite stream packets also contain the waiting time T_k. So, the node j can calculate the μ_k as the summation of the nodal delays and the link delays from the node j to k. It can also record the multiple values of μ_k for a period and calculate the σ_k. The MC node initiates a recalculation of the waiting time at each node at the regular interval because the network status can change because of node joining or departure. The instruction for recalculation is propagated downwards from the MC to the leaf of the SCDT. Thus, each MP node can update their waiting time based on the recorded means and standard deviations so that the overall profit at the MC is maximized.

6.4.2 Performance Metrics

In a CMTS session, each node generates the packets of dummy multimedia streams. These packets travel towards the MC using the SCDT. Each non-MP node forwards these packets adding its nodal delay in the packets. An MP node waits and collects the streams of its descendants until it receives all the descendants, or the waiting time expires. The MC node also does the same based on the waiting time assigned by the CMTS moderator or the user. Each stream packet is associated with a stream score. For each composition, the MC records the accumulated profits and the actual time spent to collect the profits. Suppose the MC node perform K_n compositions within a period $\bar{T}$, when the network size is n. At a stream composition t, the accumulated profit at MC is C_t^n and the time spent to collect the profit is D_t^n. The total possible profit of the system is H. So, the percentage of profit accumulation is,

$$\overline{C_t^n} = \frac{C_t^n}{H} \times 100$$

A timer scheme is better if it can provide maximum $\overline{C_t^n}$ with the lowest D_t^n. So, the following two metrics can be observed for each timer scheme to compare their performance.

(i) Average profit at network size n,

$$\boldsymbol{\overline{P}(n) = \frac{\sum_{t=1}^{K_n} \overline{C_t^n}}{K_N}} \tag{6.15}$$

(ii) Average delay for collecting profit P_n,

$$\boldsymbol{\overline{D}(n) = \frac{\sum_{t=1}^{K_n} D_t^n}{K_n}} \tag{6.16}$$

A combined metric, i.e., the Profit Collection Efficiency(PCE) can be defined as,

$$\boldsymbol{PCE = \frac{\overline{P}(n)}{\overline{D}(n)}} \tag{6.17}$$

A higher of PCE is expected from the timer schemes. The profit accumulation does not give any idea about the number of participants missed in a composition. So the Average Loss Percent(ALP) is observed, which was defined earlier and again shown below,

$$\boldsymbol{ALP_n = \frac{\sum_{i=1}^{K_n} \left\{ \frac{(n - n_i')}{n} \times 100 \right\}}{K_n} \times 100} \tag{6.18}$$

Where, n_i' is the number of participants that can deliver their streams to the MC during i^{th} composition before the waiting time expires. The number of compositions within a fixed period $\bar{T}$ is an important metric because it defines the frame rate from the MC. So, the composition rate at the MC can be defined as,

$$\boldsymbol{F_n = \frac{K_n}{\bar{T}}} \qquad (6.19)$$

We also want the F_n to be higher in a timer scheme.

6.4.3 Performance Comparison and Discussion

For the experiment, the CMTS is emulated on the cluster nodes with n=50. For the performance comparison of the timer schemes, the system is run based on a fixed topology, as shown in Figure **4.11**. The topology is formed as a peer to peer overlay network. The network size *n* grows as a new node joins the CMTS session. A node is allowed to join every 15 seconds. The system is repeatedly run for each timer scheme with W= 250, 100, and 70ms. These values are chosen because it is observed that, in the implemented system on the 50 cluster nodes, the average time taken to collect the stream scores of all nodes is between 85 to 100ms. So, one value is taken very high, one is small and the third one is near the expected average. Note that the ATS scheme does not use W. The value of $\bar{T} =$ 15 secends is used, i.e., the node joining interval. As the system runs, all the metrics defined in the previous section are calculated as analyzed.

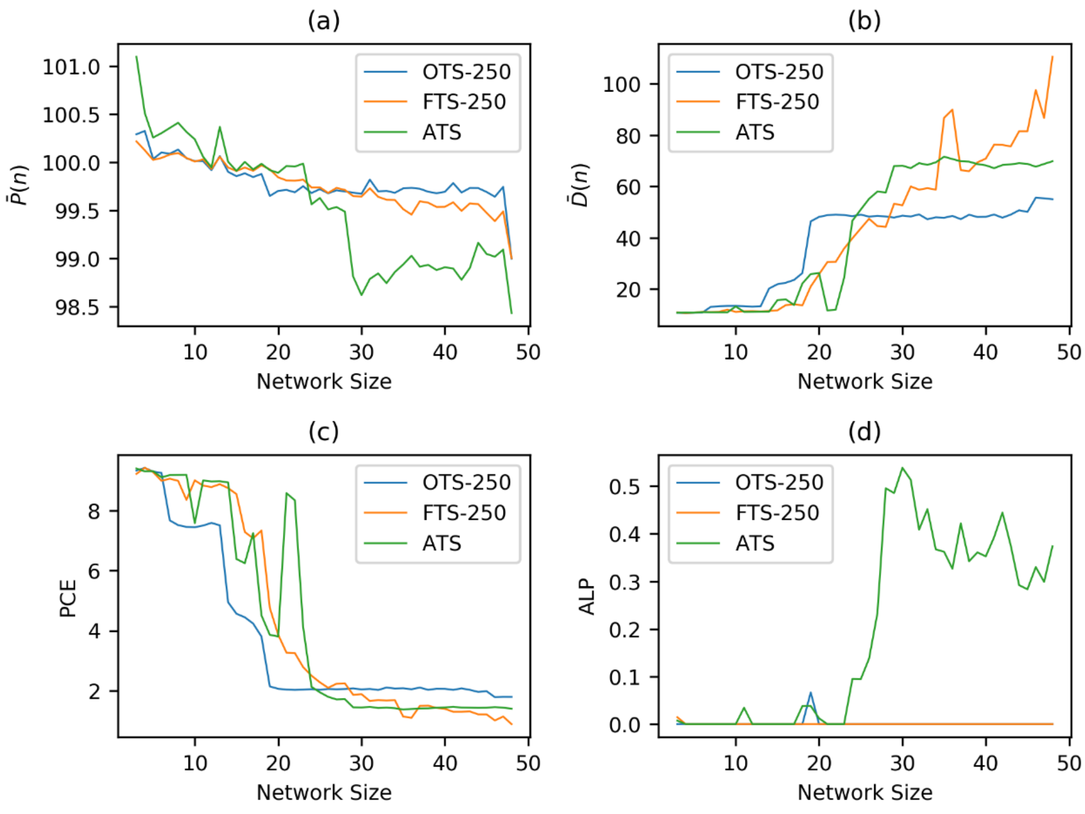


Figure 6.7 Comparison of timer schemes for W=250.

Figure **6.7** shows the comparisons of timer schemes on different metrics with W=250ms. As the waiting time is very high, the MPs and MC nodes get enough time to collect stream scores of almost all nodes (99.3%, when n=50) for both the FTS and OTS. However, the ATS does not depend on the W. In ATS, each boxing node waits for the maximum branch delay plus some margin, but it does not consider the path delay to deliver the accumulated profit to the parent, like OTS does. So, ATS can randomly miss some nodes in the composition. It can be observed from the curves in (a) that the average profit $\bar{P}(n)$ is higher for OTS than both FTS and ATS. But the profit of FTS is closer to the OTS since the boxing nodes can spend a long time to maximize the profits. Therefore, the delay $\bar{D}(n)$ is very high for FTS, (b). Note that the delay value shown in the graphs are in milliseconds. When the network size is small (less than 25), there are only a few MPs around the MC. So, at the beginning of the curves, all the schemes look similar in all metrics. But as the network grows, the number of boxing nodes do increase. So, the performance metrics are distinguishable after the network size n=25. The curves for PCE in (c) show that the PCE tends to stay high and flat against the network size for both OTS and ATS. The PCE is decreasing for the FTS because of the increasing delay metric $\bar{D}(n)$. The ALP in (c) is zero for both OTS and FTS because they spend optimal and longtime respectively to include all participants in the composition. The ATS misses some participants because of sudden increment of path delays in the network after n=20 as it responds slowly to the sudden increment. The loss tends to come down as it adjusts the waiting time.

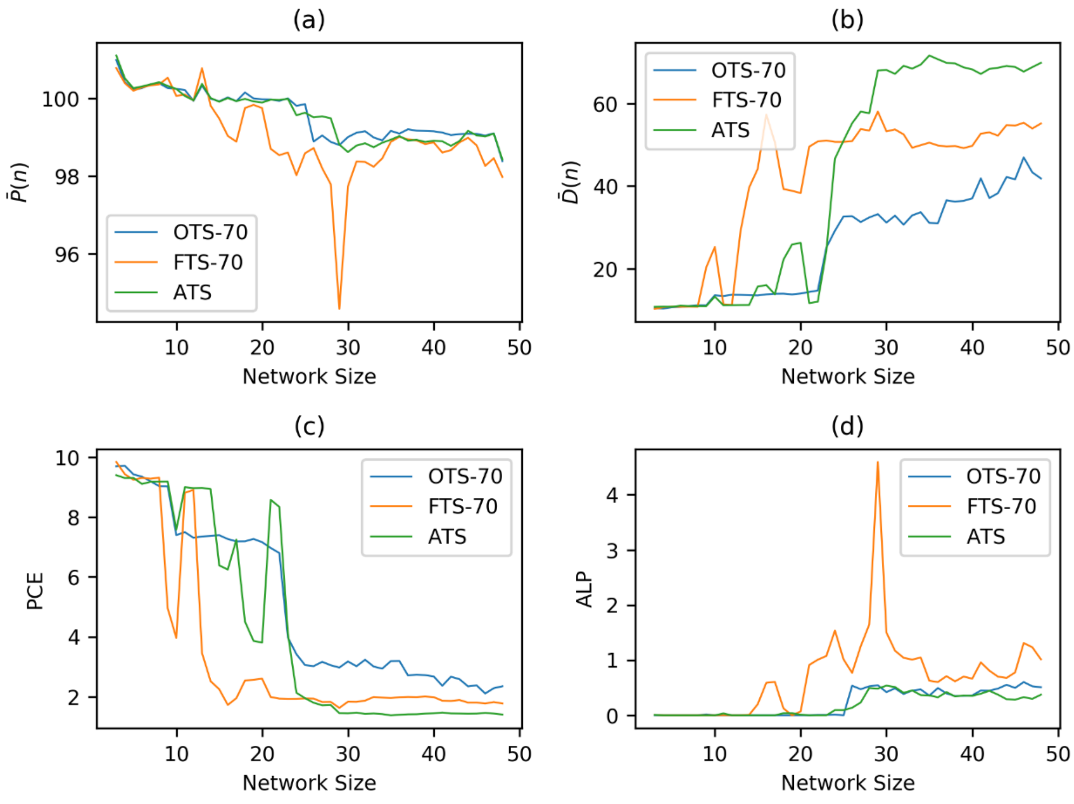


Figure 6.8 Comparison of timer schemes for W=70.

Figure **6.8** shows the performance comparison of the timer schemes for W=70. As the waiting time is slightly lower than the expected average path delays when n>25, the FTS tends to miss a lot of participants, i.e., their stream scores (a). The ATS almost matches the profit of OTS but spends a comparatively long time (nearly 70ms) when the network size is large, (b). The time is even higher than the FTS's waiting time because ATS is free to follow the network delays, not the waiting time capped by W. The time spent to collect the similar profit is significantly lower (less than 40 ms) for OTS. Therefore, the PCE is also significantly high for OTS, (c). OTS and ATS behave almost the same way to keep the loss percent ALP lower, but it is very high for FTS.

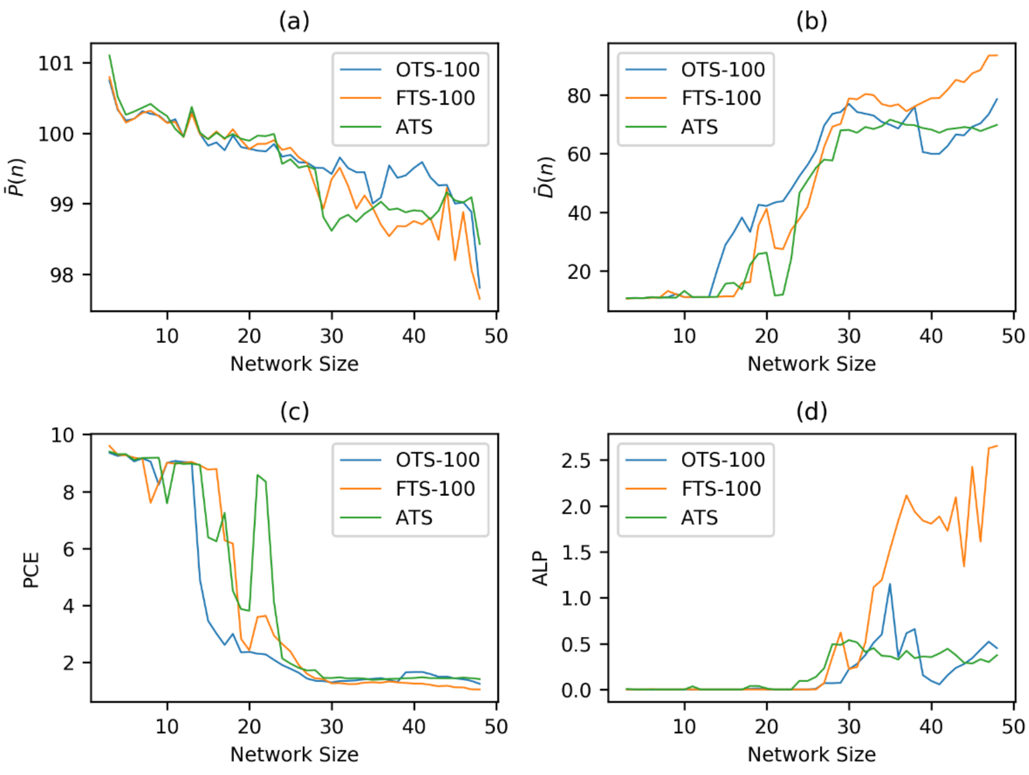


Figure 6.9 Comparison of timer schemes for W=100

The comparison of the timer schemes with W=100 is shown in Figure **6.9**. It can be observed that the OTS and ATS perform similarly, but OTS still tends to accumulate more profit. Again, FTS is worst compared to OTS and ATS for all the metrics.

In general, the OTS performs better than both ATS and FTS for all the performance metrics. ATS works fairly well compared to FTS. As we observe, the profit metric tends to go down with network size, but with high waiting time (250ms) it always stays above

99%. If the waiting time is limited to a low value, the profit metric can go below 99% for all the schemes. The delay $\overline{D}(n)$ grows linearly for FTS if the waiting is high (250ms) on the MPs because they can wait for a long time to maximize the profit. Whether the given waiting time is high or low, the OTS always yields high PCE (i.e., high profit and low delay). Thus, the OTS can help to avoid the dilemma of setting the waiting time at the MPs.

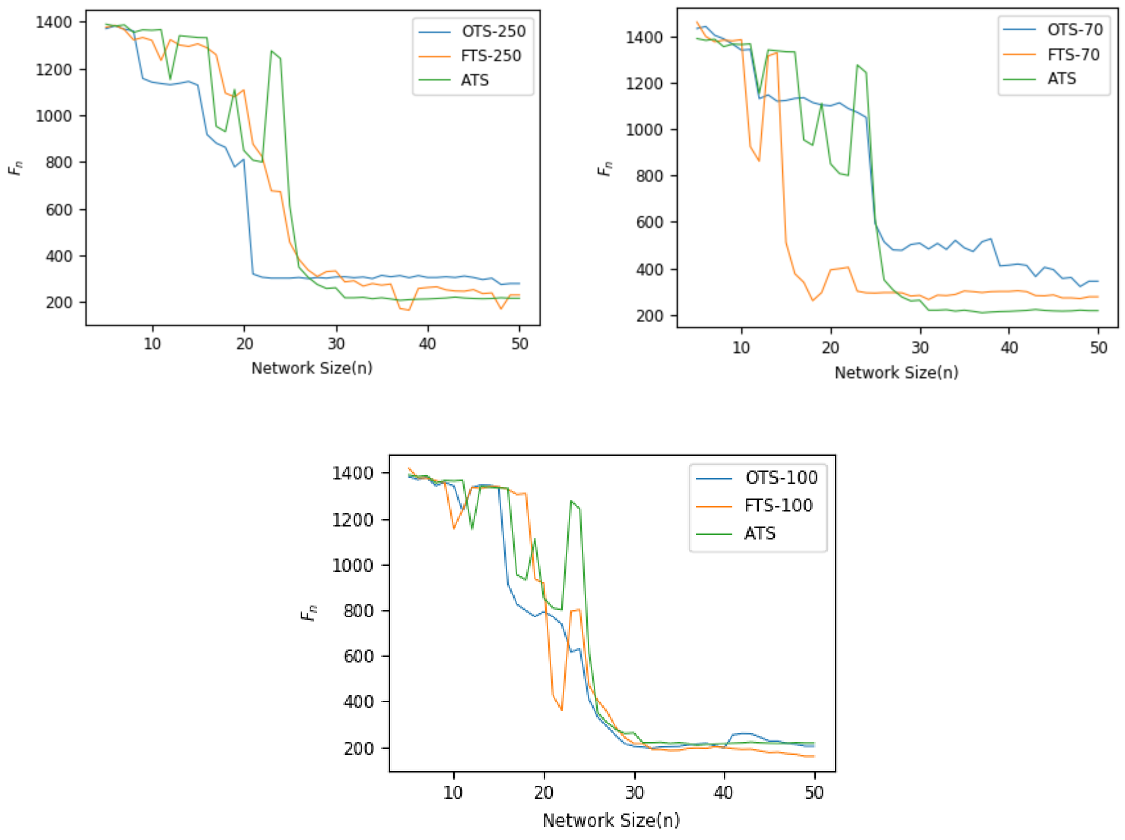


Figure 6.10 Effect of network size on the composition rate (a) W=250 (b) W=70 and (c) W=100

Figure **6.10** plots the composition rate F_n of the three schemes for comparison based on W=250, 100 and 70. It demonstrates that the composition rate (i.e., the frame rate) can go down if the network size grows. The reason is the overall growth of end-to-end delays in the overlay network. But it can be observed that the OTS keeps F_n higher compared to other waiting time schemes FTS and ATS. For CMTS, maintaining the desired frame rate with high-profit accumulation is challenging. Because the waiting time would be very low compared to the path delays from all nodes to the MC. These path delays can increase more with the network size. But, given a waiting time at MC (which is equivalent to frame interval), the OTS can maintain the expected frame rate with high-profit accumulation compared to FTS. In summary, OTS can ensure high profit $\bar{P}(n)$ and F_n with low value of $\bar{D}(n)$. To show that, another combined term, a profit score η_n^W for given waiting time W with network size n can be defined as,

$$\boldsymbol{\eta_n = \frac{\bar{P}(n)}{\bar{D}(n)} \times F_n} \tag{6.20}$$

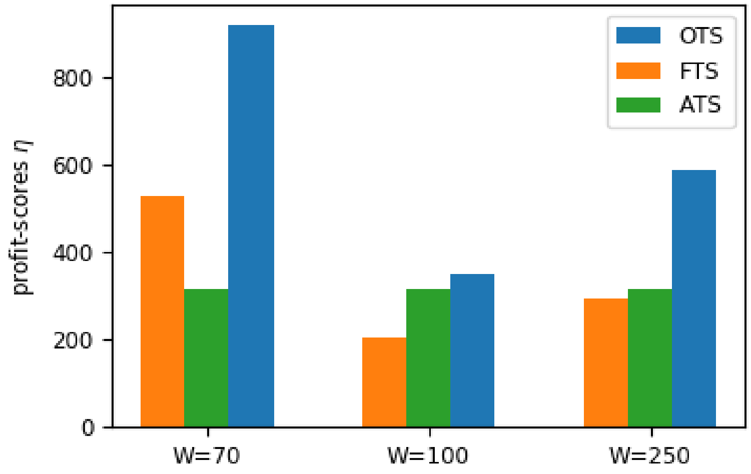


Figure 6.11 Profit scores of the timer schemes for different values of W

η_n can also be considered as the final DRDI of the system that must be maximized. For a waiting time scheme, the high value of the profit score is preferable. Figure **6.11** shows the comparison of profit scores of the three waiting time schemes for three different values of given waiting time W. It can be observed that the OTS always performs better.

However, we have also observed that if the mean value μ_j is too high for a node then the calculated timer using OTS would be zero for that node as well as all the boxing in its subtrees. The following example shows that a complete branch of the current MC (in the topology in our consideration) get the waiting time 0 because of a high mean μ_j.

```
Optimal T at node13 is 0; TK was 65, meank = 11.472 meanj =77.9648 Hop to MCU =1
Optimal T at node10 is 0; TK was 0, meank = 6.67228 meanj =58.6627 Hop to MCU =2
Optimal T at node8 is 0; TK was 0, meank = 29.1654 meanj =58.3755 Hop to MCU =4
Optimal T at node3 is 0; TK was 0, meank = 14.6644 meanj =45.1732 Hop to MCU =6
Optimal T at node1 is 0; TK was 0, meank = 7.09153 meanj =29.9477 Hop to MCU =7
```

It can exhibit some performance degradation. But because of the nodal delays, the waiting time is not completely zero. Also, because of the continuous arrival of stream packets, the streams from the branch do not entirely miss to reach the compositions.

6.5 Conclusion

In this chapter, the CMTS is further discussed, considering the stream score of the participants. An extended version of the MC election protocol is discussed that can ensure maximum possible profit accumulation within the given waiting time at MC. For distributed MCU, the MPs are placed around the MC using the algorithms presented in CHAPTER 5. For assigning waiting time at MPs, an optimal timer management scheme is discussed. The experimental results show that the optimal timer can further maximize the profit accumulation, i.e., the final DRDI within the bounded waiting time compared to some other possible waiting time management schemes.

CHAPTER 7

Conclusions and Future Work

7.1 Conclusion

Traditionally, the multi-party telepresence system is supported by one or more servers called Multipoint Control Unit(MCU). These servers are expensive, involve the third party in the system, and also bottleneck for large scale implementation. So, this dissertation presents protocols for autonomous Peer-to-Peer(P2P) implementation of a Crowd-scale Telepresence System. The protocols use multiple features from the widely adopted P2P network, Gnutella. The proposed protocols and strategies are designed based on the Principle of Distributed Computing (PDC) and the Principle of Priority-based Resource Allocation(PPRA). These principles are considered to address three of the four identified challenges of CMTS implementation, (1) Computational Challenge, (2) Temporal Challenge, and (3) Overcrowding Challenge. The fourth one is the visual challenge, which is left for future work. The PDC is used to address the first two challenges by distributing of MCU's workloads among participating peers. The MCU consists of a Multipoint Controller(MC) and one or more Multipoint Processors(MP). For distributed MCU, the optimal placement of MC and MPs in the P2P overlay network is necessary, which is time-consuming because of exponential search space. So, a phase-based design approach is considered. For optimal placement of MC, three incremental protocols, such as GAncestor, ZePoP, and ZePoP- are presented. Then, multiple methods are discussed to place the MPs around the optimal MC. For supporting the desired frame rate, two versions of progressive timer management schemes are used at MPs. The protocol ZePoP-ε is designed based on PPRA that emphasis utilizing the limited resources of the P2P network properly. Thus, PPRA is used to address the overcrowding challenge as well as the temporal challenge. It is used to design a profit-based stream collection mechanism of ZePoP-ε for maximizing a Dynamic Role and Demand based Index (DRDI) in a bounded waiting time. The proposed protocols and methods collectively ensure minimized network traffic, node hotness(load), and stream composition time with minimal drops of streams compared to the leading reported techniques

7.2 Future Work

This dissertation addresses several new issues of developing a CMTS on a P2P network. It also opens many unsolved research problems that are interesting and needed to be solved.

1. In this work, the protocols and techniques are validated by theoretical proof, simulation, and emulation. So, the immediate task can be validating them in a real system, especially using the WebRTC framework[39].
2. In practice, the waiting time is not directly given. The session chair can choose to define the quality of service in terms of composition rate, bitrate, frame rate, etc. There must be a technique to map those settings to the waiting time so that the proposed protocols can be used.
3. There is a significant task to be done for profile scoring in a real CMTS. The scoring can be done using online profiles and appearance, community voting, etc.

4. The flat network of CMTS is not scalable. The hierarchical architecture of the CMTS should be investigated. In the hierarchical model, the system should allow nested sessions or rooms inside a large room so that multiple parallel sessions can be conducted.

In addition, the following applications and related challenges can be addressed in the future.

1. *Applications*

Along with the large-scale virtual meeting applications, the proposed architecture can be considered for developing a Multimedia Assisted Disaster Management System(MADMS) for after disaster recovery. After any catastrophic disaster, text, images, stored or live videos from residents of disaster sites can be used to immediately identify the severity of damages, emergency of health services, rescue services, requirements of drinking water, food, etc. and dispatch the emergency teams automatically. However, most of the disasters knock down the cellular communication system and even before the event, such a network becomes too busy to handle overloaded communication. The broadband or WIFI networks are dead because of the damages in power distribution lines. In these situations, the idea of Smartphone Adhoc Network(SPAN) can be used to develop as large-scale peer to peer network using technologies like Wi-Fi Direct[84, 85], Apple Mupltipeer Connectivity[86], or Device to Device (D2D) communication[87] of 5G to establish temporary multimedia communication network.

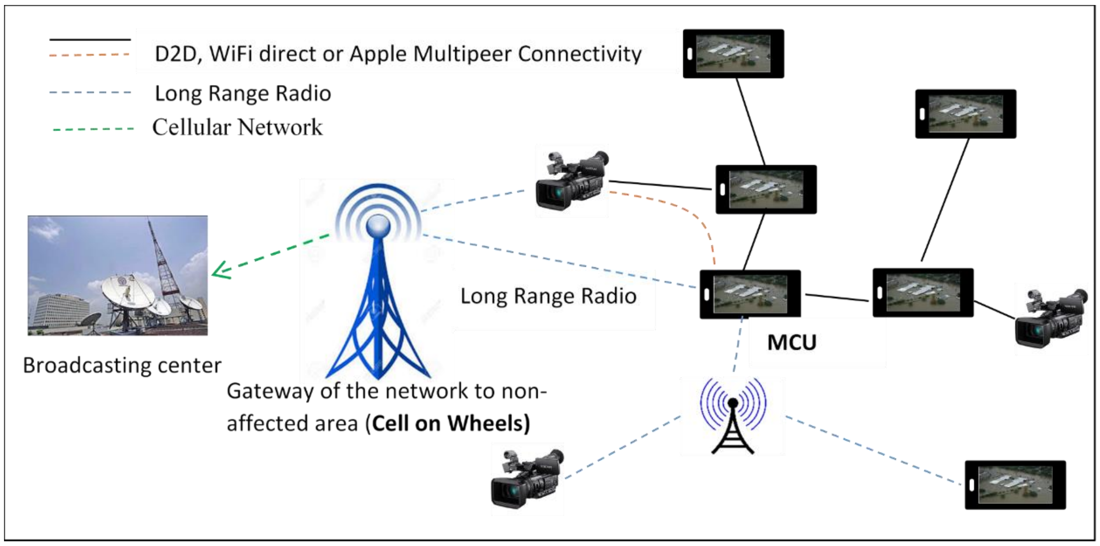

D2D, WiFi direct or Apple Multipeer Connectivity
Long Range Radio
Cellular Network
Long Range Radio
MCU
Broadcasting center
Gateway of the network to non-affected area (**Cell on Wheels)**

Figure 7.1 A network scenario for supporting MADMS

Such a network scenario is shown in Figure 7.1. The proposed algorithms in this dissertation can be directly used in such a system because this temporary network will have limited resources for carrying multimedia data.

2. *Multimedia Data Analysis to Improve Scalability*

In CMTS, the network resources can quickly be saturated, restricting additional users from joining the event. So, the multimedia data must be analyzed[88, 89] to prioritize the content so that only the highest priority contents are propagated in the network. In MADMS, the prioritization mechanism is required and it must follow the call prioritization of emergency services[90-92] so that it can be used for automatic dispatching of emergency teams.

3. *Technology Readiness for P2P Applications*

It is very challenging to build a P2P network to support a large scale system because of (a) Technology readiness level (b) lack of proper incentive mechanism so that the participants can join the network and allow to use their resources. The data processing in the system by only mobile participants would also be challenging.

4. Security and Privacy issues

In P2P applications, the same device works as both client and server, which makes the overall system vulnerable. Moreover, the detection of malicious users can be very challenging. In MADMS, the system should automatically identify false emergency calls.

4. *Efficient Video Compression for Augmented reality (especially for a fixed environment)*

In CMTS, sending visual streams from all participants can be overwhelming, even using the best compressor. However, in each frame, almost all pixels are fixed except for the eyes and mouth. We can apply machine learning techniques to perform block-level predictions of video frames by transferring only the information about the moving pixels.

REFERENCES

[1] M. MINSKY. (1980) Telepresence. Available: http://web.media.mit.edu/~minsky/papers/Telepresence.html

[2] G. M. Mair, "Telepresence-the technology and its economic and social implications," in *1997 International Symposium on Technology and Society Technology and Society at a Time of Sweeping Change. Proceedings*, 20-21 June 1997 1997, pp. 118-124, doi: 10.1109/ISTAS.1997.658870.

[3] W. A. Ijsselsteijn, "History of Telepresence," *3D Videocommunication*, pp. 5-21, 2005/07/28 2005, doi: doi:10.1002/0470022736.ch1
10.1002/0470022736.ch1.

[4] W. R. Sherman and A. B. Craig, "Chapter 1 - Introduction to Virtual Reality," in *Understanding Virtual Reality (Second Edition)*, W. R. Sherman and A. B. Craig Eds. Boston: Morgan Kaufmann, 2019, pp. 4-58.

[5] C. Luo, W. Wang, J. Tang, J. Sun, and J. Li, "A Multiparty Videoconferencing System Over an Application-Level Multicast Protocol," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1621-1632, 2007, doi: 10.1109/TMM.2007.907467.

[6] Y. Lu, Y. Zhao, F. Kuipers, and P. Van Mieghem, "Measurement Study of Multi-party Video Conferencing," in *NETWORKING 2010*, Berlin, Heidelberg, M. Crovella, L. M. Feeney, D. Rubenstein, and S. V. Raghavan, Eds., 2010// 2010: Springer Berlin Heidelberg, pp. 96-108.

[7] P. Rodríguez, A. Á, J. Salvachúa, and J. Cerviño, "dOTM: A Mechanism for Distributing Centralized Multi-party Video Conferencing in the Cloud," in *2014 International Conference on Future Internet of Things and Cloud*, 27-29 Aug. 2014 2014, pp. 61-67, doi: 10.1109/FiCloud.2014.20.

[8] Y. Wu, C. Wu, B. Li, and F. C. M. Lau, "vSkyConf: cloud-assisted multi-party mobile video conferencing," presented at the Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing, Hong Kong, China, 2013. [Online]. Available: https://doi.org/10.1145/2491266.2491273.

[9] D. Grois, D. Marpe, T. Nguyen, and O. Hadar, *Comparative assessment of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders for low-delay video applications* (SPIE Optical Engineering + Applications). SPIE, 2014.

[10] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 134-143, 2006, doi: 10.1109/MCOM.2006.1678121.

[11] J.-M. Ho, J.-C. Hu, and P. Steenkiste, "A conference gateway supporting interoperability between SIP and H.323," presented at the Proceedings of the ninth ACM international conference on Multimedia, Ottawa, Canada, 2001. [Online]. Available: https://doi.org/10.1145/500141.500204.

[12] B. Rugani and D. Caro, "Impact of COVID-19 outbreak measures of lockdown on the Italian Carbon Footprint," *Science of The Total Environment*, vol. 737, p. 139806, 2020/10/01/ 2020, doi: https://doi.org/10.1016/j.scitotenv.2020.139806.

[13] R. B. Jackson, Jones, M.W. *et al*, "Temporary reduction in daily global CO
2 emissions during the COVID-19 forced confinement," ed: Nature Climate Change, 2020.

[14] "Mesh Networking," in *https://en.wikipedia.org/wiki/Mesh_networking*, ed, [Accessed on August 2019].

[15] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proceedings First International Conference on Peer-to-Peer Computing*, 27-29 Aug. 2001 2001, pp. 101-102, doi: 10.1109/P2P.2001.990434.

[16] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 205-217, 2002, doi: 10.1145/964725.633045.

[17] T. A. Le and N. Hang, "Centralized and distributed architectures of scalable video conferencing services," in *2010 Second International Conference on Ubiquitous and Future Networks (ICUFN)*, 16-18 June 2010 2010, pp. 394-399, doi: 10.1109/ICUFN.2010.5547169.

[18] Z. Wang, J. Zhao, W. Xi, and Z. Jiang, "A Scalable P2P Video Conferencing System Based on VCStream Model," in *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, 30 May-1 June 2012 2012, pp. 77-82, doi: 10.1109/ICIS.2012.15.

[19] Willebeek-LeMair, M. H. K. D. D., Shae, and Z.-Y., "On multipoint control units for videoconferencing," presented at the Proc. 19th Conference on Local Computer Networks, Minneapolis, MN, USA, 1994.

[20] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Janus: a general purpose WebRTC gateway," presented at the Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications, Chicago, Illinois, 2014.

[21] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Performance analysis of the Janus WebRTC gateway," presented at the Proceedings of the 1st Workshop on All-Web Real-Time Systems, Bordeaux, France, 2015.

[22] L. López *et al.*, "Kurento: The WebRTC Modular Media Server," presented at the Proceedings of the 24th ACM international conference on Multimedia, Amsterdam, The Netherlands, 2016. [Online]. Available: https://doi.org/10.1145/2964284.2973798.

[23] B. Grozev, G. Politis, E. Ivov, T. Noel, and V. Singh, "Experimental Evaluation of Simulcast for WebRTC," *IEEE Communications Standards Magazine,* vol. 1, no. 2, pp. 52-59, 2017, doi: 10.1109/MCOMSTD.2017.1700009.

[24] "Selective Forwarding Unit(SFU), https://webrtcglossary.com/sfu/." https://webrtcglossary.com/sfu/ (accessed April 2020).

[25] E. André, N. L. Breton, A. Lemesle, L. Roux, and A. Gouaillard, "Comparative Study of WebRTC Open Source SFUs for Video Conferencing," in *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, 16-18 Oct. 2018 2018, pp. 1-8, doi: 10.1109/IPTCOMM.2018.8567642.

[26] B. Grozev, L. Marinov, V. Singh, and E. Ivov, "Last N: relevance-based selectivity for forwarding video in multimedia conferences," presented at the Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, Portland, Oregon, 2015. [Online]. Available: https://doi.org/10.1145/2736084.2736094.

[27] Radvision, "User Guide for SCOPIA Elite 5000 Series MCU Version 7.7," 2011.

[28] J. D. Gibson, T. Berger, T. Lookabaugh, D. Lindbergh, and R. L. Baker, *Digital compression for multimedia: principles and standards*. Morgan Kaufmann Publishers Inc., 1998.

[29] D. Tang and L. Zhang, "Audio and Video Mixing Method to Enhance WebRTC," *IEEE Access*, vol. 8, pp. 67228-67241, 2020, doi: 10.1109/ACCESS.2020.2985412.

[30] H. Toral-Cruz, J. Argaez-Xool, L. Estrada-Vargas, and D. Torres-Roman, "An Introduction to VoIP: End-to-End Elements and QoS Parameters," 2011.

[31] P. R. M.-P. Meyer, *Restoration of Motion Picture Film* (Conservation and Museology). Butterworth-Heinemann 2000.

[32] P. Antoniadis and B. L. Grand, "Incentives for resource sharing in self-organized communities: From economics to social psychology," in *2007 2nd International Conference on Digital Information Management*, 28-31 Oct. 2007 2007, vol. 2, pp. 756-761, doi: 10.1109/ICDIM.2007.4444315.

[33] A. K. e. al., "*Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal*," ed: Internet Engineering Task Force (IETF) 2018.

[34] T. K. a. R. and Manfredi. "Gnutella 0.6." http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html (accessed.

[35] G. Forum. "The Annotated Gnutella Protocol Specification v0.4." http://rfc-gnutella.sourceforge.net/developer/stable/ (accessed October 30, 2018).

[36] B. F. 2019, "BitTorrent (BTT) White Paper, v0.8.7 WORKING DRAFT [Accessed on March 2020," ed.

[37] L. Wang and J. Kangasharju, "Measuring large-scale distributed systems: case of BitTorrent Mainline DHT," in *IEEE P2P 2013 Proceedings*, 9-11 Sept. 2013 2013, pp. 1-10.

[38] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System, [Accessed on March 2020],"

[39] D. C. B. A. Bergkvist, C. Jennings, and A. Narayanan, "WebRTC 1.0: Real-time communication between browsers,"

[40] L. G. a. C. P. a. R. R. a. A. Mankin, "Large Scale Video Conferencing: A Digital Amphitheater," presented at the Proc. 8th International Conference on Distributed Multimedia Systems, 2002.

[41] S. Chen, K. Nahrstedt, and I. Gupta, "3DTI amphitheater: a manageable 3DTI environment with hierarchical stream prioritization," presented at the Proceedings of the 5th ACM Multimedia Systems Conference, Singapore, Singapore, 2014. [Online]. Available: https://doi.org/10.1145/2557642.2557654.

[42] T. K. a. R. Manfredi. "Gnutella 0.6." http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html (accessed.

[43] "BlueJeans." http://bluejeans.com/ (accessed.

[44] S. T. Chanson, A. Hui, E. Siu, I. Beier, H. Koenig, and M. Zuehlke, "OCTOPUS-a scalable global multiparty video conferencing system," in *Proceedings Eight International Conference on Computer Communications and Networks (Cat. No.99EX370)*, 11-13 Oct. 1999 1999, pp. 97-102, doi: 10.1109/ICCCN.1999.805502.

[45] M. R. Civanlar, Ö. Özkasap, and T. Çelebi, "Peer-to-peer multipoint videoconferencing on the Internet," *Signal Processing: Image Communication*, vol. 20, no. 8, pp. 743-754, 2005/09/01/ 2005, doi: https://doi.org/10.1016/j.image.2005.05.003.

[46] İ. E. Akkuş, Ö. Özkasap, and M. R. Civanlar, "Peer-to-peer multipoint video conferencing with layered video," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 137-150, 2011/01/01/ 2011, doi: https://doi.org/10.1016/j.jnca.2010.08.006.
[47] X. Yu and Z. Yu, "A Distributed Architecture of Video Conference Using P2P Technology," *JNW, Academy Publisher*, vol. 7, pp. 1852-1859, 2012.
[48] X. Z. a. C. Li, "Scalable Multipoint Videoconferencing Scheme without MCU," in *International Conference on Information Computing and Applications*, Singapore, 2013: Springer, Berlin, Heidelberg, pp. 203-211.
[49] S. Petrangeli, D. Pauwels, J. v. d. Hooft, J. Slowack, T. Wauters, and F. D. Turck, "Dynamic video bitrate adaptation for WebRTC-based remote teaching applications," 2018. [Online]. Available: https://doi.org/10.1109/NOMS.2018.8406217.
[50] S. Petrangeli, D. Pauwels, J. v. d. Hooft, T. Wauters, F. D. Turck, and J. Slowack, "Improving quality and scalability of webRTC video collaboration applications," 2018. [Online]. Available: https://doi.org/10.1145/3204949.3208109.
[51] S. Petrangeli *et al.*, "A scalable WebRTC-based framework for remote video collaboration applications," *Multimedia Tools Appl.*, vol. 78, no. 6, pp. 7419-7452, / 2019, doi: 10.1007/s11042-018-6460-0.
[52] "Centrality, https://en.wikipedia.org/wiki/Centrality#Closeness_centrality[Accessed on December 2019]." (accessed.
[53] G. L. Lann, "Distributed Systems - Towards a Formal Approach," in *IFIP Congress*, 1977.
[54] M. Garcia, "Elections in a Distributed Computing System," *IEEE Transactions on Computers*, vol. C-31, no. 1, pp. 48-59, 1982, doi: 10.1109/TC.1982.1675885.
[55] H. Abu-Amara and J. Lokre, "Election in asynchronous complete networks with intermittent link failures," *IEEE Transactions on Computers,* vol. 43, no. 7, pp. 778-788, 1994, doi: 10.1109/12.293257.
[56] G. Singh, "Leader election in the presence of link failures," *IEEE Transactions on Parallel and Distributed Systems,* vol. 7, no. 3, pp. 231-236, 1996, doi: 10.1109/71.491576.
[57] N. Santoro, *Design and Analysis of Distributed Algorithms (Wiley Series on Parallel and Distributed Computing)*. Wiley-Interscience, 2006.
[58] W. Wang and C. Y. Tang, "Distributed computation of classic and exponential closeness on tree graphs," in *2014 American Control Conference*, 4-6 June 2014 2014, pp. 2090-2095, doi: 10.1109/ACC.2014.6858727.
[59] K. You, R. Tempo, and L. Qiu, "Distributed Algorithms for Computation of Centrality Measures in Complex Networks," *IEEE Transactions on Automatic Control,* vol. 62, no. 5, pp. 2080-2094, 2017, doi: 10.1109/TAC.2016.2604373.
[60] W. Wang and C. Y. Tang, "Distributed estimation of closeness centrality," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 15-18 Dec. 2015 2015, pp. 4860-4865, doi: 10.1109/CDC.2015.7402978.
[61] M. Wu, S. Cheon, and C. Kim, "A Central Leader Election Method Using the Distance Matrix in Ad Hoc Networks," in *New Challenges for Intelligent Information and Database Systems*, N. T. Nguyen, B. Trawiński, and J. J. Jung Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 107-116.
[62] C. Kim and M. Wu, "Leader election on tree-based centrality in ad hoc networks," *Telecommunication Systems,* vol. 52, no. 2, pp. 661-670, 2013/02/01 2013, doi: 10.1007/s11235-011-9510-8.

[63] K. Mokhtarian and H. Jacobsen, "Minimum-delay overlay multicast," in *2013 Proceedings IEEE INFOCOM*, 14-19 April 2013 2013, pp. 1771-1779, doi: 10.1109/INFCOM.2013.6566975.

[64] A. e. a. Naz, "Centrality-Based Leader Election, https://projects.femto-st.fr/programmable-matter/centrality-leader-election [Accessed on December 2019]," ed.

[65] A. Naz, B. Piranda, S. C. Goldstein, and J. Bourgeois, "ABC-Center: Approximate-center election in modular robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 28 Sept.-2 Oct. 2015 2015, pp. 2951-2957, doi: 10.1109/IROS.2015.7353784.

[66] A. Naz, B. Piranda, J. Bourgeois, and S. C. Goldstein, "Electing an Approximate Center in a Huge Modular Robot with the k-BFS SumSweep Algorithm," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1-5 Oct. 2018 2018, pp. 4825-4832, doi: 10.1109/IROS.2018.8593612.

[67] A. Naz, B. Piranda, S. C. Goldstein, and J. Bourgeois, "Approximate-Centroid Election in Large-Scale Distributed Embedded Systems," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 23-25 March 2016 2016, pp. 548-556, doi: 10.1109/AINA.2016.109.

[68] U. Kang, S. Papadimitriou, J. Sun, and H. Tong, "Centralities in Large Networks: Algorithms and Observations," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, (Proceedings: Society for Industrial and Applied Mathematics, 2011, pp. 119-130.

[69] B. Grozev, G. Politis, E. Ivov, and T. Noel, "Considerations for deploying a geographically distributed video conferencing system," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 8-10 Jan. 2018 2018, pp. 357-361, doi: 10.1109/CCWC.2018.8301726.

[70] P. Rodríguez, Á. Alonso, J. Salvachúa, and J. Cerviño, "Materialising a new architecture for a distributed MCU in the Cloud," *Computer Standard & Interfaces*, vol. 44, pp. 234-242, February 2016.

[71] A. Á, I. Aguado, J. Salvachúa, and P. Rodríguez, "A Metric to Estimate Resource Use in Cloud-Based Videoconferencing Distributed Systems," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 22-24 Aug. 2016 2016, pp. 25-32, doi: 10.1109/FiCloud.2016.12.

[72] A. Á, I. Aguado, J. Salvachúa, and P. Rodríguez, "A Methodology for Designing and Evaluating Cloud Scheduling Strategies in Distributed Videoconferencing Systems," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2282-2292, 2017, doi: 10.1109/TMM.2017.2733301.

[73] D. S. Lemons, P. Langevin, and NetLibrary Inc., *An introduction to stochastic processes in physics*, Baltimore: Johns Hopkins University Press, pp.34, 2002, pp. xii, 110 p. [Online]. Available: http://ezproxy3.lhl.uab.edu/login?url=http://www.netLibrary.com/urlapi.asp?action=summary&v=1&bookid=75720.

[74] S. S. Lam and H. Liu, "Failure recovery for structured P2P networks: protocol design and performance evaluation," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 199–210, 2004, doi: 10.1145/1012888.1005712.

[75] K. Samant and S. Bhattacharyya, "Topology, search, and fault tolerance in unstructured P2P networks," in *37th Annual Hawaii International Conference on System Sciences,*

2004. Proceedings of the, 5-8 Jan. 2004 2004, p. 6 pp., doi: 10.1109/HICSS.2004.1265682.

[76] R. Iqbal, S. Shirmohammadi, and B. Hariri, "Modeling and Evaluation of a Metadata-Based Adaptive P2P Video-Streaming System," *The Computer Journal*, vol. 56, no. 5, pp. 554-572, 2012, doi: 10.1093/comjnl/bxs110.

[77] M. A. Hossain and J. I. Khan, "Dynamic MCU Placement for Video Conferencing on Peer-to-Peer Network," in *2015 IEEE International Symposium on Multimedia (ISM)*, 14-16 Dec. 2015 2015, pp. 144-147, doi: 10.1109/ISM.2015.125.

[78] Kwok-Fai Ng *et al.*, "A P2P-MCU Approach to Multi-Party Video Conference with WebRTC " *International Journal of Future Computer and Communication*, vol. 3, no. 5, October 2014.

[79] S. Y. Renjie Gu , Fan Wu, ""Distributed Machine Learning on Mobile Devices: A Survey",” arXiv preprint arXiv:1909.08329, 2019," ed, 2019.

[80] A. Bounceur, M. Bezoui, M. Lounis, R. Euler, and C. Teodorov, "A new dominating tree routing algorithm for efficient leader election in IoT networks," in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 12-15 Jan. 2018 2018, pp. 1-2, doi: 10.1109/CCNC.2018.8319292.

[81] J. I. Khan and A. U. Haque, "Computing with data non-determinism: Wait time management for peer-to-peer systems," *Comput. Commun.*, vol. 31, no. 3, pp. 629-642, 2008, doi: 10.1016/j.comcom.2007.08.047.

[82] A. Bernstein and D. Nanongkai, "Distributed exact weighted all-pairs shortest paths in near-linear time," presented at the Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, Phoenix, AZ, USA, 2019.

[83] A. U. Haque and J. I. Khan, "Cascading Multi-way Bounded Wait Timer Management for Moody and Autonomous Systems," Berlin, Heidelberg, 2011: Springer Berlin Heidelberg, in Algorithms and Architectures for Parallel Processing, pp. 24-32.

[84] Wikipedia, "Wifi direct," ed, [Accessed on October 30th, 2018].

[85] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 96-104, 2013, doi: 10.1109/MWC.2013.6549288.

[86] Apple, "Multipeer Connectivity," ed, [Accessed on October 3oth 2018].

[87] U. N. Kar and D. K. Sanyal, "An overview of device-to-device communication in cellular networks," *ICT Express*, vol. 4, no. 4, pp. 203-208, 2018/12/01/ 2018, doi: https://doi.org/10.1016/j.icte.2017.08.002.

[88] "Natural language processing, https://en.wikipedia.org/wiki/Natural_language_processing." (accessed.

[89] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 23-28 June 2014 2014, pp. 1725-1732, doi: 10.1109/CVPR.2014.223.

[90] "Emergency," in *https://en.wikipedia.org/wiki/Emergency*, ed, Accessed, April 2019.

[91] "Medical Priority Dispatch System," in *https://en.wikipedia.org/wiki/Medical_Priority_Dispatch_System*, ed, [Accessed on April 2019].

[92] U. D. o. justice. "“Differential Police Response, City of Greensboro, North Carolina., https://www.ncjrs.gov/pdffiles1/Digitization/89918NCJRS.pdf." (accessed.