ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
PH.D.

BY
BADARNEH, Osamah

SEAMLESS, RELIABLE, VIDEO MULTICAST IN WIRELESS AD HOC NETWORKS

MONTRÉAL, 20 OCTOBER 2009

ACKNOWLEDGMENT

First and foremost I would like to thank Allah. I could never have done this without the faith I have in you, Allah.

Second, I would like to thank my parents, Sami and Inam, for allowing me to realize my own potential. All the support they have provided me over the years was the greatest gift anyone has ever given me. They have given me more than I deserve.

I would like to express my sincere gratitude to Prof. M. Kadoch who has been my supervisor since the beginning of my study. He provided me with many helpful suggestions, important advice and constant encouragement during the course of this work. I also thank the other members of my thesis committee: Prof. Ahmed Elhakeem at the Department of Electrical and Computer Engineering, Concordia University, Prof. Jean-Marc Robert at the Department of Information Technology and software Engineering, Ecole de Technology Superieure, and Prof. Zbigniew Dziong at the Department of Electrical Engineering, Ecole de Technology Superieure, and Hussein Muoftah, at School of Information Technology and Engineering, University of Ottawa, for their advice and support.

My special appreciation goes to my sisters and brothers for all their love and encouragement. Finally, I am forever indebted to my beloved wife Sumaia, my little princess Lujain, and my lovely son Ammar for their understanding, endless patience and encouragement when it was most required.

LA VIDÉO MULTICAST DANS LES RÉSEAUX SANS FIL AD HOC : APPROCHE FIABLE ET PERFORMANTE

BADARNEH, Osamah

RÉSUMÉ

Un réseau sans fil *ad hoc* est un réseau dynamiquement auto-configurable et déployable sans l'existence d'aucune infrastructure préexistante ou d'entité centralisée. Les nœuds ont la liberté de se déplacer arbitrairement tout en établissant des structures de communication avec le voisinage. D'où l'utilité de tels réseaux dans le cas des communications entre des nœuds participants à un groupe dans le but d'accomplir une certaine tâche. Vu son utilité et son efficacité dans des champs d'applications orientées groupes, le multicast devient une technique essentielle pour tout type d'application tel que la distribution de la vidéo, la vidéoconférence, la diffusion des données, les opérations de secours et les champs de batailles.

Sans multicast, les flux vidéo devraient être dupliqués sur plusieurs sessions unicast, ce qui cause du gaspillage en termes de bande passante. Appliquant le multicast dans le cas des réseaux ad hoc, nous optimisons grandement l'utilisation de cette ressource. Cependant, faire de la vidéo multicast sur des canaux radio ad hoc est un vrai défi; en effet, les paquets vidéo ont des contraintes spécifiques en termes de délai et de perte. D'autre part, vu les changements topologiques fréquents occasionnés par la mobilité des nœuds dans les réseaux ad hoc, les liens font toujours face à des coupures continues et incontrôlées; ce qui dégrade drastiquement la qualité du signal vidéo reçu. D'autres défis incluent la durée de vie des batteries des nœuds mobiles, ainsi que la capacité limitée des réseaux sans fil par comparaison aux réseaux filaires.

La vidéo multicast dans les réseaux sans fil ad hoc a toujours été un axe de recherche très actif dans les années récentes. L'objectif étant d'améliorer la qualité de la vidéo reçue par l'exploitation des propriétés de résilience aux erreurs des techniques de codage à description multiple MDC appliqués sur plusieurs chemins. En d'autres mots, la vidéo MD est encodée et transmise sur deux chemins différents vers chaque nœud destination; dans le cas où un chemin est perdu, les paquets vidéo correspondant à l'autre description MD parviendront à temps chez la destination via le deuxième chemin.

Le codage en couches LC et le codage à description multiple MDC ont été proposes en tant que techniques orientées source dans le but de minimiser les effets inévitables des erreurs de transmission. Au contraire à MDC, LC encode un flux média dans deux ou plusieurs sous-flux, nommés couches; dont une couche principale représente le codage de premier niveau qui fournit une qualité vidéo de base essentielle à la compréhension de la vidéo reçue, et les autres couches sont utilisées pour fournir plusieurs niveaux d'améliorations qui permettent de raffiner la qualité globale de la vidéo étant reconstruite par la couche principale. La couche principale étant essentielle au décodage de la vidéo, il devient très important à la protéger; si cette couche est corrompue, les autres couches deviennent inutiles même si elles ont été correctement reçues. Dans MDC, les sous-flux ont la même importance dans le sens où chaque sous flux, nommé Description, peut être décodé indépendamment afin de produire un signal de qualité de base. Plus le décodeur reçoit de descriptions, meilleure est la qualité.

Envoyer du multicast vers des destinations de nature hétérogène pose des problématiques telles que l'assignation des descripteurs vidéo et la construction de l'arbre multicast. Ce qui aura un impact direct sur la satisfaction des usagers (qualité de la vidéo reçue). Dans cette thesis, nous proposons des nouvelles approches pour remédier à ce genre de dilemmes dans un contexte d'un groupe de nœuds destination hétérogènes. L'idée principale est d'appliquer la propriété d'indépendance de description de MDC à un ensemble d'arbres multicast. Cependant, plusieurs questions peuvent avoir lieu : Combien d'arbres multicast faut-il construire? Combien de vidéo MD devrait-on assigner? Serait-il souhaitable de construire préalablement les arbres multicast et d'assigner par la suite les descriptions vidéo ou bien vaut-il mieux commencer par l'assignation des descriptions MD suivie de la construction des arbres? Et finalement, faut-il appliquer ces mécanismes d'une façon centralisée ou distribuée?

Pour répondre à ces questions, nous proposons de différents algorithmes pour construire plusieurs arbres multicast, et assigner les descriptions MD. Ces algorithmes sont : MDC en série, MDC distribué et MDC centralisé. En effet, MDC en série construit plusieurs chemins vers chaque destination, et assigne une description vidéo différente à chacun d'eux; ensuite, il construit plusieurs arbres multicast. Le nom de MDC en série réfère au fait que ce dernier construise plusieurs chemins et assigne un MD vidéo à chaque destination d'une façon indépendante et centralisée. Le MDC distribué assigne le MD vidéo et construit les arbres multicast en parallèle et d'une façon distribuée. Dans MDC centralisé, l'assignation des MD vidéo ainsi que la construction des arbres sont réalisées d'une façon centralisée; dans ce cas, MDC centralisé construit en premier plusieurs arbres multicast, et assigne par la suite une description vidéo différente à chaque arbre. Nous évaluons et comparons les algorithmes proposés dans des conditions réseau différentes; par exemple, la taille du réseau et la taille des groupes multicast. Les résultats des simulations viennent pour valider les principes proposés et montrer une meilleure satisfaction de la part des usagers. En plus, ces résultats démontrent que MDC permet de réaliser une meilleure satisfaction usagère par comparaison à LC; pourtant ceci présente un cout supplémentaire minime en termes de nombre de nœuds retransmetteurs, d'utilisation de bande passante, et d'agrégation des délais sur les arbres.

En outre, nous utilisons les algorithmes proposés afin de concevoir des protocoles multicast pour transmettre la vidéo multicast dans un réseau sans fil ad hoc. Plus spécifiquement, nous proposons quatre protocoles, nommés MDMTR centralisé, MDMTR séquentiel, MDMTR distribué, et MDMTR détecteur de voisinage. Ces protocoles prennent en considération plusieurs métriques, comme la mobilité des nœuds, la maintenance des arbres multicast, et la gestion des adhésions à des groupes multicast. Nous évaluons les performances de ces protocoles et les comparons ensemble dans de différentes conditions du réseau ad hoc; par exemple, la taille des groupes multicast et la mobilité des nœuds. Les résultats des simulations ont validé que ces protocoles démontraient de meilleures performances par comparaison avec d’autres protocoles existants dans la littérature.

Mots clés: diffusion groupée, réseaux sans fil ad hoc, lecture vidéo en transit, codage à description multiple

SEAMLESS, RELIABLE, VIDEO MULTICAST IN WIRELESS AD HOC NETWORKS

BADARNEH, Osamah

ABSTRACT

A wireless ad hoc network is a self-organized and dynamically reconfigurable wireless network without central administration and wired infrastructure. Nodes in a wireless ad hoc network can instantly establish a communication structure while each node moves in an arbitrary manner. A wireless ad hoc network is useful for mobile nodes working in a group to accomplish certain tasks. On the other hand, multicast is a very useful and efficient means of supporting group-oriented applications. Multicast is an essential technology for many applications such as video distribution and group video conferencing, data dissemination, disaster relief and battlefield.

Video multicasting over wireless ad hoc networks is bandwidth-efficient compared to multiple unicast sessions. However, video multicasting poses great challenges over wireless ad hoc networks. Video packets are both delay and loss sensitive. In addition, due to nodes mobility, the topology of wireless ad hoc networks is frequently changed. As a result, the established links are continuously broken, causing quality loss and interruption in the received video signal. Other challenges include limited battery life of wireless nodes and lower wireless network capacity compared to wired networks.

Video multicast over wireless ad hoc networks has been an active area in recent years. The main objective of these studies is to improve the quality of the received video by exploiting the error resilience properties of Multiple Description Coding (MDC) along with multiple paths. In other words, MD video is encoded and transmitted over two different paths to each destination node. If only one path is broken, packets corresponding to the other description on the other path can still arrive at the destination node on time.

Layered Coding (LC) and Multiple Description Coding (MDC) have been proposed as video source coding techniques that are robust against inevitable transmission errors. In contrast to MDC, LC encodes a media source into two or more sub-streams, known as *layers*, one base layer and several enhancement layers. The base layer can be decoded to provide a basic quality of the received video while the enhancement layers are mainly used to refine the quality of the video that is reconstructed from the base layer. If the base layer is corrupted, the enhancement layers become useless, even if they are received correctly. Therefore, the base layer is critically important and is usually highly protected. For MDC, however, these sub-streams are of equal importance in the sense that each sub-stream, also called a description, can be decoded *independently* to produce a signal of basic quality. When more descriptions are received, the decoder can gradually increase the quality.

One main problem of video multicasting for heterogeneous destinations is the assignment of video descriptions and the construction of multicast tree. However, the assignment of MD video and the construction of multicast tree can greatly affect the user satisfaction (i.e., affect the quality of the received video). In this thesis, we introduce novel approaches to improve the user satisfaction for a set of heterogeneous multicast destinations. The main idea of our approaches is to employ the *independent-description* property of MDC along with multiple multicast trees. However, many questions are raised: How multiple multicast trees should be constructed? And how MD video should be assigned? Is it better to construct multiple multicast trees first and then assign the video descriptions? Or is it better to assign the video descriptions first and we then construct multiple multicast trees? Should we perform that in a distributed manner or in a centralized one?

To answer these questions, we propose different algorithms to construct multiple multicast trees and to assign MD video. The proposed algorithms are: Serial MDC, Distributed MDC, and Centralized MDC. Serial MDC constructs multiple paths, to each destination, and assigns a different video description to each of them. After that, it constructs multiple multicast trees. Distributed MDC assigns MD video and constructs multiple multicast trees in parallel and in distributed fashion. In Centralized MDC, the assignment of MD video and the construction of multiple multicast trees are performed in a centralized manner. However, Centralized MDC first constructs multiple multicast trees and then assigns different video description to each multicast tree. We evaluate and compare our proposed algorithms under different network conditions. For example, Network size, and multicast group size. Simulation results demonstrate that, indeed, the way of multicast trees construction and the assignment of MD video can greatly affect the user satisfaction. In addition, simulation results show that MDC can achieve higher user satisfaction compared to LC with a small cost in terms of number of pure forwarders nodes, bandwidth utilization, and aggregate tree delay.

Furthermore, we use our proposed algorithms to develop different multicast protocols for video multicast over wireless ad hoc networks. Specifically, we propose four protocols, namely, Centralized MDMTR (Multiple Disjoint Multicast Trees Routing), Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR protocols. These protocols take many issues into consideration, rejoining and joining a multicast group, multicast trees maintenance, and mobility of nodes, for example. We evaluate the performance of our proposed protocols and compare them under different network conditions. For example, multicast group size, and mobility of nodes. Simulation results demonstrate that our protocols perform well compared to other protocols in the literature.

Keywords: Multicast, Wireless ad hoc networks, Video streaming, Multiple description coding

TABLE OF CONTESTS

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADMR	Adaptive Demand Driven Multicast Routing
AODV	Ad Hoc on Demand Distance Vector
API	Application Programming Interfaces
CCA	Common Code Assignment
CDMA	Code Division Multiple Access
DCT	Discrete Cosine Transform
GA	Genetic Algorithm
GL	Group Leader
IP	Internet Protocol
LC	Layered Coding
MAC	Medium Access Control
MACT	Multicast Activation message
MANET	Mobile Ad Hoc Network
MAODV	Multicast Ad Hoc on Demand Distance Vector
MD	Multiple Description
MDC	Multiple Description Coding
MDMTR	Multiple Disjoint Multicast Trees Routing
MDTMR	Multiple Disjoint Trees Multicast Routing
Mgroup-Hop	Multicast Group Hop Count
MNDP	Multiple Node-Disjoint Paths
MSMT	Multiple Steiner Minimum Tree
MSPT	Multiple Shortest Path Tree
MT-MAODV	Multiple Tree Multicast Ad Hoc On-Demand Distance Vector
MZRP	Zone Routing Protocol for Multicasting
NP	Nondeterministic Polynomial time
ODMRP	On Demand Multicast Routing Protocol
QoS	Quality of Service
RAM	Rate-Adaptive Multicast
RCA	Receiver-based Code Assignment

RDVMR	Robust Demand-driven Video Multicast Routing
RREP	Route Reply
RREQ	Route Request
SNP/VQR	Scalable, Noncausal Prediction with Vector Quantization and conditional Replenishment
TCA	Transmitter-based Code Assignment
TDMA	Time Division Multiple Access
TTL	Time-To-Live
VC	Virtual Circuit

INTRODUCTION

Wireless networks are becoming increasingly popular as they provide millions of people every day access to information at anytime and from anywhere. Hence, wireless networks have the potential to enable ubiquitous communications. A Conventional wireless network is usually supported by centralized administration and wired infrastructure. In contrast, a wireless ad hoc network is self-organized and dynamically reconfigurable without centralized infrastructure and wired infrastructure. Nodes in a wireless ad hoc network can instantly establish a communication structure while each node moves in an arbitrary manner. Thus a wireless ad hoc network is useful for mobile nodes working in a group to accomplish a certain task. Multicast is a very useful and efficient mean of supporting group-oriented applications. Multicast is an essential technology for many applications such as video distribution and group video conferencing, data dissemination, disaster relief and battlefield.

As wireless ad hoc networks emerge as a promising solution for future ubiquitous communications, there is a compelling demand to support various types of video applications in these networks considering the increasing volume of video traffic of the cellular network and Internet. Video multicasting over wireless ad hoc networks is bandwidth-efficient compared to multiple unicast sessions. However, video multicasting poses great challenges over wireless ad hoc networks. Unlike data packets, video packets are delay and loss sensitive. In addition, due to nodes mobility, the topology of wireless ad hoc networks is frequently changed. As a result, the established links are continuously broken, causing quality loss and interruption in the received video signal. Other challenges include limited battery life of wireless nodes and lower wireless network capacity compared to wired networks.

In this thesis, we focus on video multicast routing over wireless ad hoc networks. Although our simulations run over wireless ad hoc network, our proposed algorithms for distributing Multiple Description (MD) video and constructing multiple multicast trees can be applied as well to other networks such as wired networks, mesh networks, multihop wireless networks, and private virtual networks, etc.

Problem Statement

Mobile nodes, in ad hoc wireless networks, are expected to be heterogeneous with a set of multicast destinations greatly differing in their end devices and Quality of Service (QoS) requirements. However, existing video multicast routing protocols do not consider heterogeneity of destinations. Furthermore, they are developed under the assumption that destinations wish to receive all the video information sent by a multicast source. The main objective of these protocols is to improve the quality of the received video by using a new technology of video coding, known as Multiple Description Coding (MDC), along with multiple paths. In MDC, the raw video is encoded into multiple *independent* descriptions (e.g. two descriptions) and each description is transmitted over a different path to destination node. If one path is broken, packets corresponding to the other description on the other path can still arrive to the destination on time.

Our problem of video multicast can be formulated as follows. Given a wireless ad hoc network with a multicast source and a set of multicast destination nodes, each of them has a preference number of MD video. Then, construct multiple multicast trees such that the number of assigned MD video to each destination is maximized. Clearly, there are two main problems, namely, multicast trees construction and MD video assignment. However, these two problems raise many questions. Should the multicast trees and the assignment of MD video performed in a distributed manner or in a centralized one? Which one of them should be performed before, the construction of multicast trees or the assignment of MD video? Should multiple paths construction and MD video assignment be performed to each destination alone? Or should multicast trees construction consider all destinations at the same time and then assign the MD video? Furthermore, what about trees disjointness? Should multicast trees (multiple paths to each destination) be totally node-disjointed trees? Or should be non-disjointed trees?

However, in wireless ad hoc networks as nodes in the network move or as wireless transmission conditions change, some nodes (e.g. forwarders or destination nodes) may become disconnected from the multicast forwarding tree of the multicast group. Therefore, when a broken link is detected between two nodes then which node is responsible to repair the broken link? Maintaining the multicast group can be achieved by either the *Soft-State* approach or the *Hard-State* approach. In the Soft-State approach, the multicast group membership and associated routes are refreshed periodically (*proactively*) by the flooding of control packets, whereas in the Hard-State approach broken links are reconfigured locally. Moreover, when a destination node becomes disconnected from one or more multicast tree, how should it rejoin the multicast tree? In addition, if a new node wishes to join the multicast group, how should it perform the joining process?

Objectives

In this thesis we will focus on video multicast, over wireless ad hoc network, to a set of heterogeneous multicast destinations. Our main objective is to maximize the number of MD (Multiple Description) video assigned to the destination node. As a result, the quality of the received video will be improved. To achieve our objective, we propose to deploy the *independent-description* property of MDC along with multiple paths (multiple multicast trees).

Due to the nature of MDC, the less correlated packet drop between multicast trees, the more robust the video multicast. The robustness problem we refer to is that of the disjointness between multicast trees. We assume the network is lightly loaded, i.e., mobility and poor channel conditions rather than congestion are major causes of packet drop. In this case, if multicast trees do not share any middle nodes, packet drop over multicast trees are independent. Therefore, multicast trees should be robust against nodes mobility (the higher robustness, the more efficient and reliable the multicast trees). There are two main types of multicast trees, namely *node-disjoint*, and *non-disjoin*. A set of node-disjoint trees have no common nodes except the source and the destinations. In contrast, non-disjoint trees can have links, and therefore nodes, in common. Note that node-disjointness implies link-disjointness. Provided they are available, node-disjoint trees are usually preferred because they utilize the most available network resources, therefore they are the most fault-tolerant. In principle, when an intermediate node in a set of node-disjoint trees fails, only the tree containing the failed node is affected, and therefore packets corresponding to a particular description will be lost temporarily while other packets corresponding to the other descriptions, on the other trees, will arrive the destinations on time . On the other hand, non-disjoints trees offer the least degree of fault-tolerance. A node failure may bring down multiple trees. Therefore, packets corresponding to multiple descriptions will be lost. As a result, some destinations may not receive temporarily any video signal. Hence, we consider node-disjoint multicast trees. Furthermore, to reduce the packet drop (i.e., to increase the reliability), the paths between the multicast source and each destination should contains a minimum number of forwarders nodes. The lower number of forwarding nodes, the more reliable the protocol. Hence, the shortest path algorithm is deployed during the construction of multiple paths to each destination nodes. Moreover, during joining/rejoining and repairing links failure paths with minimum hop counts will be selected.

For multimedia transmission (voice or video), the connection-oriented approach is usually taken. Such connections do not permit interferences or competitions from others in order to guarantee the delay and jitter. For such a high QoS guarantee, the multimedia transmission needs an independent channel with its exclusive bandwidth. We thus consider CDMA over TDMA channel model at the MAC sub-layer. In this model Virtual Circuits (VCs) are created (bit pipe) between a multicast source and destination nodes. Once VCs are created, video packets are delivered, to destinations, in a seamless way.

The scarce bandwidth availability in wireless ad hoc networks demands minimal control overhead for the multicast session. Efficient multicast routing protocols are expected to provide a fair number of control packets transmitted through the network relative to the number of data packets reaching their destination intact. One way to reduce control overhead is to deploy on-demand (reactive) routing approach. A reactive routing protocol has no prior knowledge of the network topology but finds a route to a given destination on demand. On the other hand, a proactive routing protocol tries to always maintain a complete updated picture of the network topology. Reactive routing protocol are normally preferred when nodes are highly mobile, when only a subset of nodes are communicating at any one time, and when communication sessions last for relatively long times. On the other hand, proactive routing protocols are preferred for lower levels of mobility and when communication is random and sporadic. Another way to reduce control overhead is to deploy local repair (hard-state) approach for repairing broken links. In the hard-state approach, the control packets are transmitted only when a link breaks. On the other hand, in the soft-state approach control packets are flooded periodically to refresh the routes. We thus consider on-demand multicast routing along with local repair approach.

Finally, if a multicast routing protocol needs the support of a particular routing protocol, then it is difficult for the multicast protocol to work in heterogeneous networks. Hence, it is desirable to have a multicast routing protocol which is independent of any specific unicast routing protocol. Thus, our proposed protocols are independent (i.e., they do not depend on any specific unicast routing protocol).

Research Methodology

Layered Coding (LC) and MDC have been proposed as source coding techniques that are robust against inevitable transmission errors. In contrast to MDC, LC encodes a media source into two or more sub-streams, known as layers, one base layer and several enhancement layers. The base layer can be decoded to provide a basic quality of the received video while the enhancement layers are mainly used to refine the quality of the video that is reconstructed from the base layer. If the base layer is corrupted, the enhancement layers become useless, even if they are received correctly. Therefore, the base layer is critically important and is usually highly protected. For MDC, however, these sub-streams are of equal importance in the sense that each sub-stream, also called a description, can be decoded *independently* to produce a signal of basic quality. When more descriptions are received, the decoder can gradually increase the quality.

In this thesis, we, first, present a comparison between the aforementioned video source coding techniques in terms of different metrics, namely, user's satisfaction, number of pure forwarders nodes, bandwidth utilization, and aggregate tree delay. Specifically, we propose three algorithms for assigning MD video and constructing multiple multicast trees, namely, Serial MDC, Distributed MDC, and Centralized MDC. To validate our proposed algorithms, we compare them to the algorithm proposed in (Chen, Chan et Li, 2004), we refer to as Chen-LC. This algorithm deploys LC as a source coding technique. Furthermore, it assigns multiple video layers to a group of heterogeneous destinations and constructs multiple multicast trees. The main objective of this algorithm is to maximize the "success rate" (number of assigned video layers to the destinations). We carry out the comparison under different condition, such as, varying number of destinations and varying number of nodes in the network (network size). Our comparison shows the effectiveness of our proposed algorithms in terms of user's satisfaction with a small cost in bandwidth utilization, number of pure forwarders nodes, and aggregate tree delay. Hence, we select MDC as a source coding technique.

The aforementioned algorithms consider the *independent-property* of MDC during the assignment of MD video. We, further, propose two algorithms, namely, Multiple Shortest Path Trees (MSPT) and Multiple Steiner Minimum Trees (MSMT), for constructing multiple multicast trees and assigning MD video. Both algorithms aim at improving the user satisfaction for a set of heterogeneous destinations. However, these algorithms do not consider the *independent-description* property of MDC during the assignment of MD video. Both algorithms construct different types of multiple multicast trees, namely, node-disjoint, Hybrid-I, and Hybrid-II trees. Simulation results demonstrate that MSMT can slightly achieve a higher user satisfaction compared to MSPT.

Second, we evaluate the performance of our proposed protocols and compare them to Serial MDTMR (Multiple Disjoint Tree Multicast routing) protocol (Wei et Zakhor, 2007). The MDC is used for video coding. Without loss of generality, we consider that the raw video is encoded into two descriptions. Each description represents a QoS of level one. While the two descriptions represent a QoS of level two.

We conduct the comparison under different network conditions. For example, varying number of destinations, and varying node mobility. Simulation results demonstrates that our proposed algorithms outperform Serial MDTMR in several metrics, user's satisfaction, number of pure forwarder nodes, ratio of bad frames, number of bad periods, normalized packet overhead, and control overhead.

Thesis Contribution

In this thesis, we design and develop a framework for video multicast, for heterogeneous destinations, over wireless ad hoc network. Specifically, we propose four on-demand multicast routing protocols for video distribution over wireless ad hoc networks. In the process, we develop novel (1) algorithms for assigning MD video to a set of destinations and (2) algorithms for constructing multiple multicast trees for video streaming. Each protocol employs different algorithms for constructing multiple multicast trees and assigning MD video. These protocols are: Centralized Multiple Disjoint Multicast Trees Routing (Centralized MDMTR) protocol, Sequential MDMTR protocol which is a variant of Centralized MDMTR, Distributed MDMTR protocol, and Neighbor-aware MDMTR protocol which is a variant of Distributed MDMTR. However, this is the first framework for video multicast for heterogeneous destinations over wireless ad hoc network that deploys the *independent-description* property of MDC to improve the user satisfaction and as a result the quality of the received video is improved.

Centralized MDMTR and Sequential MDMTR construct multiple multicast trees and assign MD video in a centralized way. Centralized MDMTR considers the *independent-description* property of MDC during the assignment of MD video. In contrast, Sequential MDMTR sequentially assigns MD video to the destination. This means that all destination nodes should be assigned the first description. After that, destinations that require another description (they have QoS of level two) will be assigned the second description. In order to minimize routing overhead and construction delay, we further propose Distributed MDMTR and Neighbor-aware MDMTR.

Both protocols, Distributed MDMTR and Neighbor-aware MDMTR, assign MD video and constructs multiple multicast trees in a distributed way. These protocols take into account several issues such as: (i) joining/leaving and rejoining a multicast group and (ii) multicast trees maintenance.

Thesis Outline

The rest of the thesis is organized as follows: Chapter 1 gives a brief description about wireless ad hoc networks, traditional multicast protocols and challenges of video over ad hoc networks.

In chapter 2, we present a detail description of the related work. In this chapter, we mainly focus on video multicast over wireless ad hoc network. However, we present in details some existing multicast routing protocols, namely, MAODV, ODMRP, and ADMR. We only discuss these protocols, because existing video multicast protocols are based on them. In addition, we introduce the existing video multicast protocols and we discuss there advantages and disadvantages. In addition we present the general architecture of multiple paths (multiple trees) video streaming over wireless ad hoc networks.

In chapter 3, we study the problem of multiple multicast trees construction and the assignment of MD video. We then propose three algorithms for that purpose. The first algorithm constructs multiple multicast trees and assigns MD video in a serial way. The second algorithm randomly assigns MD video and it then constructs multiple multicast trees accordingly. The last one constructs multiple multicast trees and it then assigns MD video accordingly. The *independent-description* property of MDC is deployed during the assignment of MD video. In this chapter, we compare MDC with LC to show that MDC achieve higher user's satisfaction with a small cost in terms of utilized bandwidth and number of pure forwarding nodes.

In chapter 4, we present in details our proposed protocols for video multicast streaming. The protocols are: (Centralized MDMTR) protocol, Sequential MDMTR protocol which is a variant of centralized MDMTR, Distributed MDMTR protocol, and Neighbor-aware MDMTR protocol which is a variant of Distributed MDMTR. The functionality of each protocol is described in details. In addition, we describe in detail the construction of multiple multicast trees and the assignment of MD video process, multicast trees maintenance, and joining/leaving and rejoining a multicast group. Finally, we conduct several simulations to evaluate our proposed protocols

In chapter five, we conclude the thesis and we present some future directions.

CHAPITRE 1

INTRODUCTION

1.1 Introduction

Multicasting plays a crucial role in many applications of wireless ad hoc networks. It can significantly improve the performance of these networks, the channel capacity and battery power of which are limited. In the past couple of years, a number of multicast routing protocols have been proposed. In spite of being designed for the same networks, these protocols are based on different design principles and have different functional features when they are applied to the multicast problem. The interested reader can refer to (Badarneh et Kadoch, 2009) for more details about multicast routing protocols in wireless ad hoc networks. The authors present a coherent survey of existing multicasting solutions for wireless ad hoc networks. In addition, they present various classifications of the current multicast routing protocols, discuss their operational features, along with their advantages and limitations, and provide a comparison of their characteristics according to several distinct features and performance parameters.

The rest of the chapter is organized as follows: in Section 1.2, we introduce a brief description of wireless ad hoc networks. In Section 1.3, we describe design issues of current multicast protocols. Section 1.4 discusses the challenges of video streaming over ad hoc networks. In Section 1.5 we give a brief introduction about existing researches on video multicast over wireless ad hoc networks.

1.2 Wireless Ad Hoc Networks

A wireless ad hoc network comprises either fixed or mobile nodes connected wirelessly without the support of any fixed infrastructure or central administration. The nodes are self-organized and can be deployed “on the fly” anywhere, any time to support a particular purpose. Two nodes can communicate if they are within each other’s transmission range; otherwise, intermediate nodes can serve as relays (routers) (multi-hop routing). These networks have several salient features: rapid deployment, robustness, flexibility, inherent mobility support, highly dynamic network topology (device mobility, changing properties of the wireless channel, i.e. fading, multipath propagation, and partitioning and merging of ad hoc networks are possible), the limited battery power of mobile devices, limited capacity, and asymmetric/unidirectional links. Wireless ad hoc networks are envisioned to support advanced applications such as military operations (formations of soldiers, tanks, planes), civil applications (e.g. audio and video conferencing, sport events, telematics applications (traffic)), disaster situations (e.g. emergency and rescue operations, national crises, earthquakes, fires, floods), and integration with cellular systems (Perkins, 2000; Toh, 2002). Figure 1.1 demonstrates a typical wireless ad hoc network.

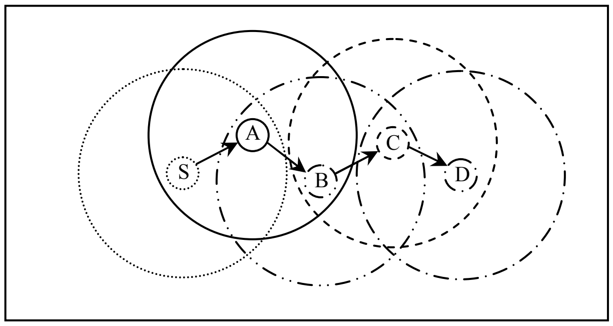


Figure 1.1 Demonstration of a wireless ad hoc network: Node S communicates with node D over multi-hop path. The large circles represent each node’s transmission range.

1.3 Multicast Routing Protocol Design: Issues and Challenges

While *unicast* routing involves a pair of communicating nodes, in *multicast*, multiple nodes can be involved in a communication session simultaneously. Figure 1.2 depicts a typical multicast session in a wireless ad hoc network. A multicast source S transmits data to a group of multicast destinations. Destination nodes can be also intermediate (forwarder) nodes as well as leaves nodes.

Multicasting in wireless ad hoc networks is much more complex than in wired networks and faces several challenges: Multicast group members move, which precludes the use of a fixed infrastructure multicast topology, wireless channel characteristics can vary over time, and there are restrictions on node energy and capacity. The multicast protocols proposed for wired networks cannot be directly ported to wireless ad hoc networks due to the lack of mechanisms available for handling the frequent link breakages and route changes, or due to the differing characteristics of the two networks. Chiang et al. has proposed many mechanisms for adapting the wired multicast protocols to wireless ad hoc networks (Chiang, Gerla et Zhang, 1998b; Chiang et Gerla, 1997; Chiang, Gerla et Zhang, 1997; Gerla, Chiang et Zhang, 1999) Simulation results in (Chiang, Gerla et Zhang, 1998b; Chiang et Gerla, 1997; Chiang, Gerla et Zhang, 1997; Gerla, Chiang et Zhang, 1999) show an increase in control packet overhead and a rapid decrease in throughput with increased node mobility. In addition, the simulation results show that these approaches indicate the need to explore alternative multicast strategies.

Several multicast routing protocols for wireless ad hoc networks have been proposed and evaluated (Badarneh et Kadoch, 2009). These protocols are based on different design principles and have different operational features when they are applied to the multicast problem. The properties favored depend on the protocol.

The particular features of wireless ad hoc networks make the design of a multicast routing protocol a challenging one. These protocols must deal with a number of issues, including, but not limited to, high dynamic topology, limited and variable capacity, limited energy resources, a high bit error rate, a multi-hop topology, and the hidden terminal problem. The requirements of existing and future multicast routing protocols and the issues associated with these protocols that should be taken into consideration are listed below:

- **Topology, Mobility, and Robustness:** In wireless ad hoc networks, nodes are free to move anywhere, any time, and at different speeds. The random and continued movement of the nodes leads to a highly dynamic topology, especially in a high-mobility environment. A multicast routing protocol should be robust enough to react quickly to the mobility of the nodes and should adapt to topological changes in order to avoid dropping a data packet during the multicast session, which would create a low packet delivery ratio (PDR: the number of non duplicate data packets successfully delivered to each destination versus the number of data packets supposed to be received at each destination). It is very important to minimize control overhead while creating and maintaining the multicast group topology, especially in an environment with limited capacity.

- **Capacity and Efficiency:** Unlike wired networks, wireless ad hoc networks are characterized by scant capacity caused by the noise and interference inherent in wireless transmission and multi-path fading. Efficient multicast routing protocols are expected to provide a fair number of control packets transmitted through the network relative to the number of data packets reaching their destination intact, and methods to improve and increase the available capacity need to be considered.

- **Energy Consumption:** Energy efficiency is an important consideration in such an environment. Nodes in wireless ad hoc networks rely on limited battery power for their energy. Energy-saving techniques aimed at minimizing the total power consumption of all nodes in the multicast group (minimize the number of nodes used to establish multicast connectivity, minimize the number of overhead controls, and so on) and at maximizing the multicast life span should be considered.

- **Quality of Service and Resource Management:** Providing QoS assurance is one of the greatest challenges in designing algorithms for wireless ad hoc network multicasts. Multicast routing protocols should be able to reserve different network resources to achieve QoS requirements such as, capacity, delay, delay jitter, and packet loss. It is very difficult to meet all QoS requirements at the same time because of the peculiarities of ad hoc networks. Even if this is done, the protocol will be very complex (many routing tables, high control overhead, high energy consumption, etc.). As a result, doing so will not be suitable for these networks with their scarce resources, and resource management and adaptive QoS methods are more convenient than reservation methods for wireless ad hoc networks.

- **Security and Reliability:** Security provisioning is a crucial issue in wireless ad hoc network multicasting due to the broadcast nature of this type of network, the existence of a wireless medium, and the lack of any centralized infrastructure. This makes wireless ad hoc networks vulnerable to eavesdropping, interference, spoofing, etc. Multicast routing protocols should take this into account, especially in some applications such as military (battlefield) operations, national crises, and emergency operations. Reliability is particularly important in multicasting, especially in these applications, and it becomes more difficult to deliver reliable data to group members whose topology varies. A reliable multicasting design depends on the answers to three questions (Ilyas, 2002): By whom are the errors detected? How are error messages signaled? How are missing packets retransmitted?

- **Scalability*:*** A multicast routing protocol should be able to provide an acceptable level of service in a network with a large number of nodes. It is very important to take into account the nondeterministic characteristics (power and capacity limitations, random mobility, etc.) of the wireless ad hoc network environment in coping with this issue.

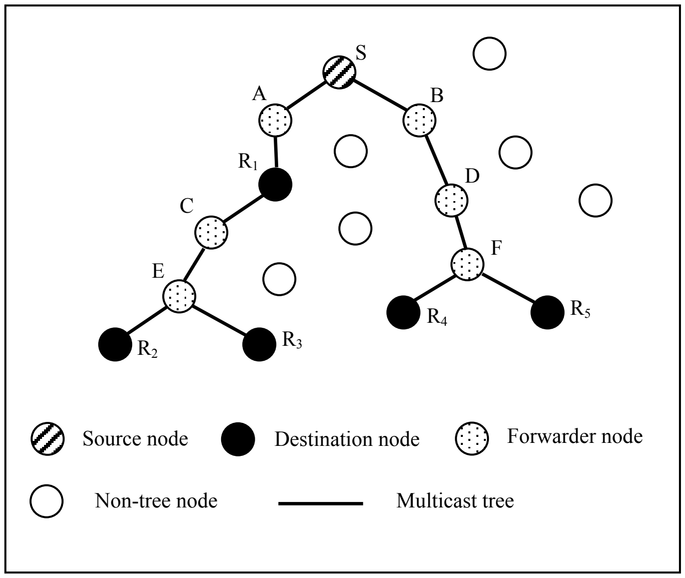


Figure 1.2 A typical multicast session in a wireless ad hoc network.

1.4 Challenges of Video Streaming Over Wireless Ad Hoc Networks

Video multicast over wireless ad hoc networks is bandwidth-efficient compared to multiple unicast sessions. However, video multicast poses great challenges over wireless ad hoc networks.

In contrast to data packets, video packets are delay and loss sensitive. Even though some packet loss is tolerable, the quality of reconstructed video or audio will be impaired and errors will propagate to consecutive frames because of the dependency introduced among frames belonging to one group of pictures at the encoder (Wang et Zhu, 1998). Unlike data packets, late arriving video packets are useless to the video decoder. Thus, the retransmission techniques, which guarantee the successful transmission of data packets, are not applicable to video communication applications.

There are additional challenges for supporting video communication over wireless ad hoc networks. First, nodes in a wireless ad hoc network are allowed to move in an uncontrolled manner. Such node mobility results in a highly dynamic network with rapid topological changes. Thus the established connection routes between sources and destinations are likely to be broken during video transmission, causing interruptions, freezes, or jerkiness in the received video signal. Second, the underlying wireless channel provides much lower and more variable bandwidth than wired networks. An end-to-end connection route in wireless ad hoc networks generally consists of multiple wireless links, which makes the available bandwidth per node even lower. Third, a wireless link usually has high transmission error rate because of shadowing, fading, path loss, and interference from other transmitting users. An end-to-end route in wireless ad hoc networks has higher error rate compared to single hop wireless connections in a wireless network with an infrastructure, since it is the concatenation of multiple wireless links. Fourth, mobile devices running on batteries have limited energy supply, and video encoding and transmission typically consume a great deal of power. These constraints and challenges, in combination with the delay and loss sensitive nature of video applications, make video communication over wireless ad hoc networks a challenging problem.

Nodes in ad hoc network are expected to be heterogeneous with a set of multicast destinations greatly differing in their end devices and QoS requirements. Thus, when a source multicasts a broadband signal, not all intended destinations are willing to receive or capable of receiving the complete signal. However, providing video services to heterogeneous destinations in wireless ad hoc networks is particularly challenging as (i) the heterogeneous destinations processing and display capabilities typically prevent destinations from processing and displaying the same encoded video information, and (ii) wireless connections typically suffer from bandwidth variability and transmission errors.

1.5 Related Work on Video Over Wireless Ad Hoc Networks

Video multicast over wireless ad hoc networks has been studied in the recent years (Agrawal, Reddy et Murthy, 2006; Anirudh, Reddy et Murthy, 2006; Chow et Ishii, 2008; He, Lee et Guan, 2009; Mao et al., 2006; Wei et Zakhor, 2007; Wei et Zakhor, 2004). We will describe these researches in detail in the next chapter. Most of researches focus on multipath video streaming. A popular approach in multipath video streaming is to use a new source coding technique referred to as MDC. The recent advances in MDC made it a promising technology for multimedia applications in wireless ad hoc networks. MDC has been proposed as an alternative of the Layered Coding (LC) technique. In contrast to LC, MDC fragments a single media stream into *independent* bit-streams, where the multiple bit-streams are referred to as multiple descriptions, each roughly of equal importance, such that any received description can be independently decoded to give a usable reproduction of the original signal. We refer to this property as *independent-description property* of MDC (Badarneh, Kadoch et Elhakeem, 2008; 2009a; 2009b; Badarneh et al., 2009). The quality of the decoded signal is commensurate with the number of received descriptions. The idea of MDC is to provide error resilience to media streams. Since an arbitrary subset of descriptions can be used to decode the original stream, network congestion or packet loss, which is common in best- effort networks such as the Internet, will not interrupt the stream but only cause a temporary loss of quality. The quality of a stream can be expected to be roughly proportional to data rate sustained by the receiver (Goyal, 2001a).

The main objective of these studies is to improve the quality of the received video by exploiting the error resilience properties of MDC along with multiple paths. In other words, MD video are encoded and transmitted over two different paths to each destination node. If only one path is broken, packets corresponding to the other description on the other path can still arrive at the destination node on time.

CHAPITRE 2

VIDEO MULTICAST ROUTING OVER WIRELESS AD HOC NETWORKS

2.1 Introduction

The nature of wireless ad hoc networks with a shared medium, the lack of a central authority for scheduling packets, a potentially low bandwidth, and mobility of the nodes make it a challenge to offer connections of a quality sufficient for real-time applications and multimedia (voice or video). A potentially promising approach to this problem is to establish multiple paths between the source and the destination of a traffic stream and to use coding schemes that take advantage of the existence of multiple paths. One such coding scheme is MDC, in which a video stream is encoded into multiple sub-streams (*descriptions*). Receiving only one description is already acceptable for the destination, but correctly and timely receiving additional descriptions improve the quality. There is no priority among the descriptions, they all play equivalent roles.

In this chapter we will focus on video multicast routing over wireless ad hoc network. Many researchers have studied the video multicast over wireless ad hoc networks. The main objective of these studies is to improve the quality of the received video by exploiting the error resilience properties of MDC along with multiple paths. In other words, MD video are encoded and transmitted over two different paths to each destination node. If only one path is broken, packets corresponding to the other description on the other path can still arrive at the destination node on time.
The rest of this chapter is organized as follows. In Section 2.2, we describe the general form of MDC. In Section 2.3, we introduce a general architecture for multipath video streaming over wireless ad hoc networks. In Section 2.4, we describe the related work. Finally, Section 2.5 concludes the chapter.

2.2 Multiple Description Coding (MDC)

Conventionally, video coding aims to encode the raw video into a single stream of video frames suitable for storage and transmission. Generally, these coding schemes use motion-compensation prediction between frames, block-Discrete Cosine Transform (DCT) for error prediction, and entropy coding. One major problem of conventional video coding is the propagation of errors when a single frame error affects not only the current frame but also all subsequent frames, until a frame with new prediction is received. In order to combat this problem, Layered Coding (LC) is introduced to encode the video stream into two layers, i.e. a base layer and an enhancement layer. In this case, the base layer alone is enough to reconstruct the original video but at lower quality. The enhancement layer is used to improve the video quality further. In normal circumstance, both layers of video are sent, but under a heavy traffic load, only the base layer is sent. This method is more robust but the same problem may occur if the base layer is lost, since the enhancement layer is totally useless by itself. Consequently, with the MDC scheme it is proposed to generate several equally important and totally *independent* video descriptions. Each description can be decoded *independently* to obtain the original video at low but acceptable quality, and every additional description received improves the quality further (Goyal, 2001b). MDC is more error-resilient than the conventional methods and it has been proven to outperform conventional coding schemes in most lossy networks (Fitzek et al., 2004).

Figure 2.1 shows the general form of MDC coding. Initially, the raw video stream is divided into several sub streams prior to the encoding stage. One of the simplest ways to do this is the per-frame allocation scheme as shown in Figure 1(a) (Fitzek et al., 2005). Next, these sub-streams are encoded independently to obtain several video descriptions. There is no constraint on the selection of the type of encoder used; either a standard video encoder such as H.264 (Software, 2009) and MPEG4, or a specifically designed MD encoder (Fitzek et al., 2005) may be used. As compared to a single video stream, a higher bandwidth is required to accommodate the MD encoded video due to the larger difference between neighboring frames within each sub-stream. This results in a lower compression efficiency and the resulting data size is larger. In (Fitzek et al., 2004), a comprehensive analysis of video encoded by the MDC scheme shows that the average frame size increases as the number of descriptions increases. The size of this increase depends on the content of the video; normally, high motion video produces a larger increase as the number of descriptions increases. The decoding process is illustrated in Figure 2.1(b). Each description is decoded separately and the reconstructed video sub-streams are merged for viewing purpose. At this stage, a compensation scheme may be used to replace any missing frame with the one displayed previously. This compensation scheme can reduce distortion during display of the displaying.

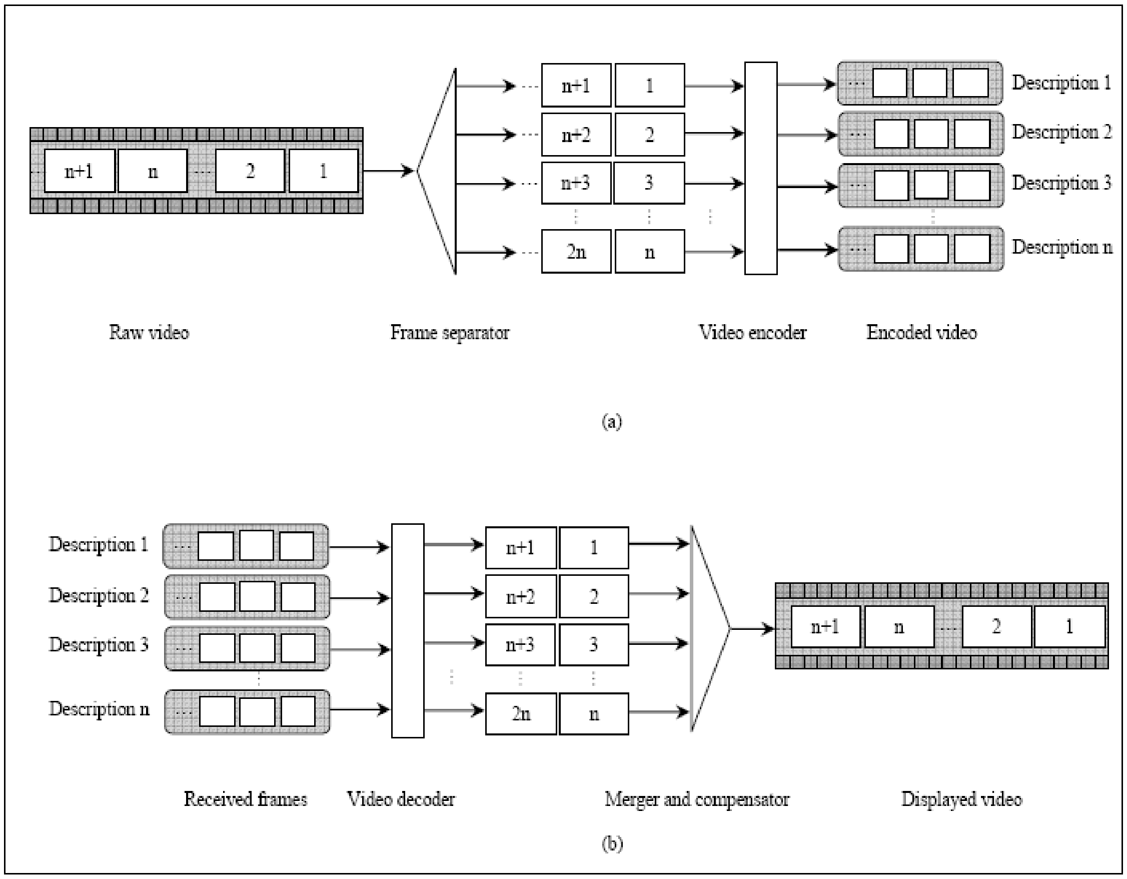


Figure 2.1 Multiple Description Coding (MDC):
(a) Encoding process; (b) Decoding process.
Taken from Chow (2007, p. 2048).

2.3 Video Streaming with Multiple Paths: General Architecture

The general architecture for multiple trees video streaming over wireless ad hoc networks is shown in Figure 2.2 (Mao et al., 2003). The video encoder generates MD video, using MDC, for transmitting them over wireless ad hoc network. The basic idea behind MDC is to generate multiple compressed descriptions of the media in such a way that a reasonable reconstruction is achieved if any one of the multiple description is available for decoding, and the reconstruction quality is improved if more descriptions are available (Tang et Zakhor, 2002; Wang et Lin, 2002). The main advantage of MDC over layered coding is that no one specific description is needed in order to render the remaining descriptions useful. Because all the descriptions of MDC are equally important, it is not necessary to protect one stream over another. Also, because each description alone can provide a low but acceptable quality, no retransmission is required, making MDC more suitable for applications with stringent delay requirements, e.g. interactive video applications. Traffic allocator decides how to distribute video packets into multiple paths in order to maximize the received video quality. For MDC video, since different descriptions are equally important, we simply distribute packets representing different descriptions into different paths. The decoder will attempt to reconstruct a video sequence from the received descriptions.

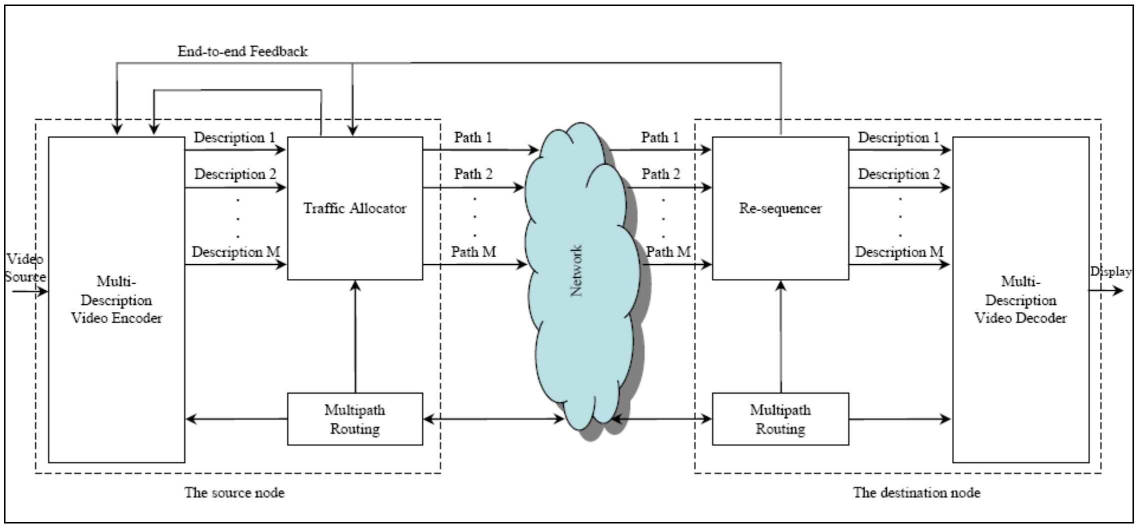


Figure 2.2 General architecture of multi-path video streaming over wireless ad hoc networks.
Taken from Mao (2003, p. 1722)

2.4 Multicast Routing Protocols in Wireless Ad Hoc Networks

This section describes the operation of some existing multicast routing protocols. These protocols are Multicast On-Demand Distance Vector Routing (MAODV) protocol, On-Demand Multicast Routing Protocol (ODMRP), and Adaptive Demand-Driven Multicast Routing (ADMR) protocol. We describe only these protocols because existing video multicast protocols are built based on them. For more details about multicast routing protocols in wireless ad hoc networks, interested reader may refer to (Badarneh et Kadoch, 2009).

2.4.1 MAODV: Multicast Ad hoc On-Demand Distance Vector Routing Protocol

The MAODV (Royer et Perkins, 2000) protocol is extended from AODV (Perkins, 1997). It maintains a shared tree for each multicast group, which consists only of destinations and relays (forwarding nodes). It determines a multicast route on demand by using a broadcast route discovery mechanism. The first member of a multicast group becomes the leader of that group. The multicast group leader is responsible for maintaining the multicast group sequence number and broadcasting this number to the multicast group. This is done through a group Hello message. Nodes use the group Hello information to update their *Request Table*.

Route Request Message Generation

A node sends a Route Request (RREQ) message when it wishes to join a multicast group, or when it has data to send to a multicast group and it does not have a route to that group. The RREQ may be either broadcast or unicast depending on the information available at the source node. If the source node has a record of another node (the multicast group leader) previously requesting a route to that multicast group, and if the source node has a valid route to that node, it includes an extension field containing the IP address of the group leader and unicasts the RREQ along the known path to the group leader. Otherwise, if the source does not know who the group leader is or if it does not have a route to the group leader, it broadcasts the request.

Figure 2.3 (a) illustrates the propagation of a broadcast RREQ. Only a member of the desired multicast tree (i.e., a router for the group) may respond to a join RREQ. If the RREQ is not a join request, any node with a fresh enough route to the multicast group may respond. If a node receives a join RREQ for a multicast group of which it is not a member, or if it receives a RREQ and it does not have a route to that group, it rebroadcasts the RREQ to its neighbors.

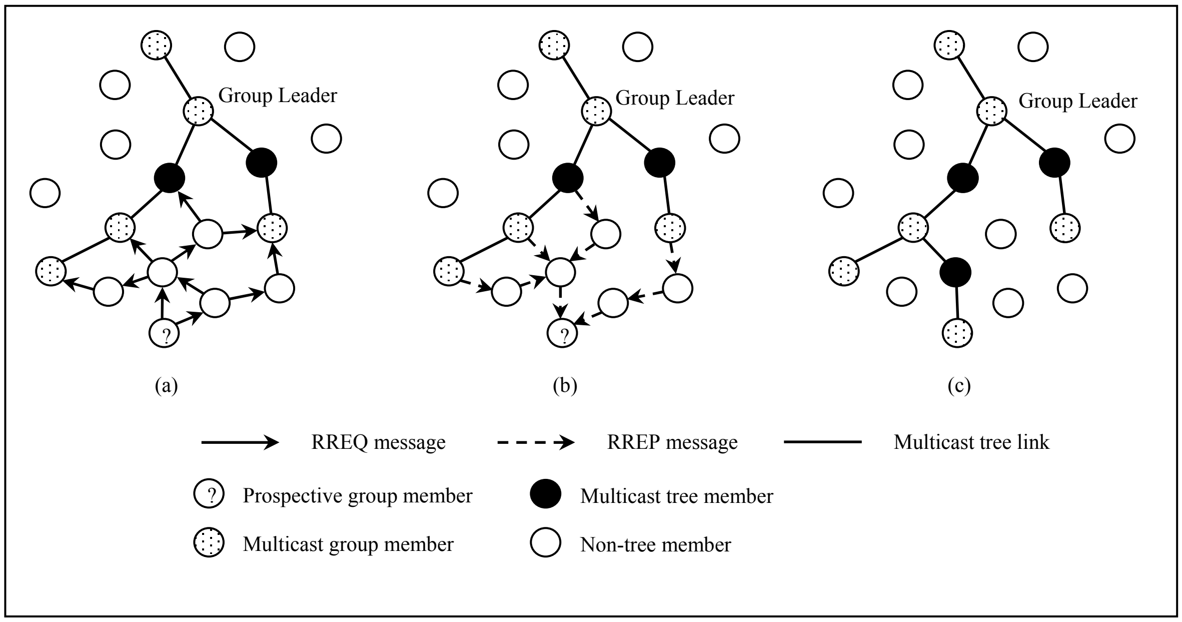


Figure 2.3 Multicast join operation in MAODV: (a) RREQ message propagation. (b) RREPs sent back to source. (c) Multicast tree branch addition.
Taken from Royer (2000, p. 4)

If the source node does not receive a RREP before timing out, it broadcasts another RREQ with *Broadcast-ID* increased by one. If it does not receive a RREP to this RREQ, it continues broadcasting route requests up to *rreq_retries* (e.g., *rreq_retries* = 2) total rebroadcasts. After this number of attempts, it can be assumed that either the multicast group is unreachable, or there are no other members of that multicast group in its connected portion of the network. In this case, the node becomes the multicast group leader, and initializes the group sequence number (i.e., sets equal to 1). If the original RREQ is unicast to the group leader and a RREP is not received, all further RREQs are broadcast, because it is possible that either the group leader is unreachable or that the node specified in the unicast RREQ is no longer the group leader.

Nodes receiving a join RREQ check their request table for an entry for the requested multicast group. If there is no entry for the multicast group, the node enters the multicast group address, together with the IP address of the requesting node, in its request table. If there is no previous entry for the group, the requesting node may become the group leader. A node wishing to join a multicast group consults its request table to determine the group leader.

Route Reply Message Generation

If a node receives a join RREQ for a multicast group, it may reply if it is a router for the multicast group's tree and its recorded sequence number for the multicast group is at least as great as that contained in the RREQ. Additionally, the group leader can always reply to a join RREQ for its multicast group. The responding node updates its route and multicast route tables by placing the requesting node's next hop information in the tables, and then generates a RREP. The node then unicasts the RREP back to the node indicated by the *Source-Address* field of the received RREQ. As nodes along the path to the source node receive the RREP, they add both a route table and a multicast route table entry for the node from which they received the RREP, thereby creating the forward path. Nodes continue forwarding the RREP back towards the source node. Figure 2.3(b) illustrates the path of the RREPs to the source node.

Group Hello Messages

The first member of the multicast group becomes the leader for that group. This node remains the group leader until it decides to leave the group, or until two partitions of the multicast tree merge. The multicast group leader is responsible for maintaining the multicast group sequence number and for disseminating this number to the multicast group.

Periodically, the group leader broadcasts a Group Hello message. The Group Hello message is an unsolicited RREP with a TTL (Time-To-Live) greater than the diameter of the network, so that it is propagated across the entire network.

Nodes use the Group Hello information to update their request table. When a node receives the Group Hello, it checks its request table for an entry for the advertised multicast group. If the table does not contain an entry for that group, the node enters the group and group leader IP addresses. Nodes that are members of the multicast tree use the Group Hello to update their current distance from the group leader. The Group Hello is also used for merging partitioned multicast trees.

Multicast Tree Maintenance

Because the network consists of mobile nodes, links on the multicast tree are likely to break. Link breakages must be repaired in a timely manner to maximize multicast group connectivity. Multicast tree maintenance can be divided into three main categories: (1) selecting and activating the link to be added to the tree when a new node joins the group, (2) pruning the tree when a node decides to leave the group, and (3) repairing a broken link. Repair involves reestablishing branches when a link fails and reconnecting the tree after a network partition.

Multicast Route Activation: When a source node broadcasts a RREQ for a multicast group, it often receives more than one reply. Because each of the RREPs sets up a potential addition to the multicast tree, one and only one of the RREPs must be selected as the next hop. In this way, only one branch is added to the tree, and loops are thereby avoided. This is accomplished as follows. The source node waits for a short time (*rte-discovery-timeout*) after sending the RREQ before selecting. During this time period, the node keeps the received route with the greatest sequence number and the shortest number of hops to the nearest member of the multicast tree; it disregards other routes. At the end of this period, it enables the selected next hop in its multicast route table, and then unicasts a Multicast Activation (MACT) message to this selected next hop.

The next hop, on receiving the MACT message, likewise enables the entry for the source node in its multicast route table. If this node is a member of the multicast tree, it does not propagate the MACT any further. However, if this node is not a member of the multicast tree, it will have received one or more RREPs from its neighbors. It keeps the best next hop for its route to the multicast group, unicasts a MACT to that next hop, and enables the corresponding entry in its multicast route table. This process continues until the node that originated the RREP (because it was already a member of the tree) is reached. Nodes that had generated or forwarded RREPs delete the entry for the requesting node if they do not receive a MACT activating their route after pre-specified timeout (*mtree-build*). Figure 2.3(c) illustrates a multicast tree created in the described manner.

The MACT message ensures that the multicast tree does not have multiple paths to any tree node. Nodes only forward data packets along activated routes in their multicast route table. This prevents the possibility of data packets being delivered to a source node by multiple next hops before a MACT message is received.

Pruning: During normal network operation, a multicast group member may decide to terminate its membership in the multicast group. If the node is not a leaf node of the tree, it may revoke its member status but it must continue to serve as a router for the tree. Otherwise, if the node is a leaf node, it may prune itself from the tree by using the MACT message (P-flag (prune) of the MACT is set). A leaf node necessarily has only one next hop for the multicast group, so it unicasts the MACT message to that next hop. After sending the message, the node removes all information for the multicast group from its multicast route table. The next hop, on receiving the MACT, notes the P-flag, and consequently deletes the entry for the sender node from its multicast route table. If this node is itself not a member of the multicast group, and if the pruning of the other node has made it a leaf node, it can similarly prune itself from the tree by the method described. Tree branch pruning terminates when either a multicast group member or a non-leaf node is reached.

Repairing Broken Links: Multicast group tree links may break due to node mobility or route expiration timers. Unlike in the unicast scenario, however, a link breakage necessarily triggers route reconstruction because of the necessity of keeping the multicast group members connected during the lifetime of the group.

Nodes promiscuously record the reception of any neighbor's transmission. A link breakage is detected if no packets are received from the neighbor in the time (*hellointerval* x (1 + *allowed_hello_loss*)). If a neighbor transmits other packets during that time, the neighbor is no longer obligated to transmit any Hello packets because the other packets serve the purpose of signaling its presence. The neighbor is also expected to forward any data packets received to their next hop(s) within a time known as *retransmit-time*. Failing to receive any transmissions from a neighbor will cause the expiration of the route timer associated with that route.

When a link breakage is detected, the node downstream of the break (i.e., the node that is further from the multicast group leader) is responsible for repairing the broken link. This distinction is made because, if both nodes tried to repair the link, it is possible they would establish different paths and thus form a loop. The downstream node initiates the repair by broadcasting a RREQ to its neighbors. The only nodes which may reply to a RREQ with the *Mgroup-Hop* extension are nodes that are at least as close to the group leader as indicated by this field, or the group leader itself. This prevents nodes on the same side of the break as the initiating node from responding, thereby ensuring a new route to the group leader is found.

Because the node with which the initiating node lost contact is likely to still be nearby, the initial TTL value of the RREQ is set to a small value. In this way, the effects of the link breakage can be localized. If no RREP is received within a pre-specified timeout (*rte_discovery_timeout*), all successive RREQs (up to *rreq_rtries* additional attempts) are broadcast across the network. Any node that is a part of the multicast tree and that has a fresh enough multicast group sequence number and a hop count to the multicast group leader smaller than that indicated by the *Mgroup-Hop* field can respond to the RREQ by unicasting a RREP.

If no RREP is received at the source node after *rreq_retries* attempts, it can be assumed that the network has become partitioned and the tree cannot (at this time) be reconnected. In this scenario, the partition of the tree that is downstream of the break is left without a group leader. A new group leader must be chosen. This occurs in one of two ways. If the node that initiated the route rebuilding is a multicast group member, it becomes the new multicast group leader. On the other hand, if it was not a group member and has only one next hop for the tree, it prunes itself from the tree by sending its next hop a MACT message with the P-flag set. On receiving the MACT, the node notes that the message came from its link to the group leader. This indicates that a network partition has occurred and that the next hop has pruned itself from the tree. If this node is a multicast group member, it becomes the new group leader. Otherwise, it also prunes itself from the tree, and this process will continue until a multicast group member is reached.

In the event that, the node that initiated the rebuilding is not a group member and has more than one next hop, it cannot prune itself from the tree because doing so would leave the tree partitioned. Instead, it selects the first of its next hops and unicasts a MACT with the GL-flag (group leader) set. This flag indicates that the next group member to receive the MACT should become the new group leader. Hence, if the next hop receiving this message is a group member, it becomes the group leader. Otherwise, if it is not a group member, it similarly selects one of its next hops and unicasts a MACT with the GL-flag set. This process continues until a multicast group member is reached.

After becoming the new multicast group leader, the node broadcasts a Group Hello across its connected part of the network (partition). This message has the U-flag (update) set, indicating that it is the new group leader and all nodes should update their multicast route table and request table information accordingly. After a multicast tree link breakage is discovered, if the node upstream of the break is a not a group member, and if the link breakage causes this node to become a leaf node, it sets a timer and waits for the tree branch to be reestablished through it. If it does not receive a MACT from a downstream node within *route-expiration* time, either another node was chosen as the next hop on the tree, or the network has become partitioned and the link could not be reestablished. In either case, it prunes itself from the tree in the manner described previously.

Reconnecting Partitioned Trees

After the multicast tree becomes disconnected due to a network partition, there are two group leaders. If the partitions reconnect, a node eventually receives a Group Hello for the multicast group that contains group leader information that differs from the information it already has. If this node is a member of the multicast group, and if it is a member of the partition whose group leader has the lower IP address, it can initiate the reconnection of the multicast tree. The node must already be a member of the group in order to minimize the number of tree branches of the group, and its group leader must have the lower IP address so that only one of the group leaders attempt to rebuild the tree, thereby avoiding loops.

If a node meets the above criteria, it unicasts a RREQ with the R-flag (repair) set to its group leader. The R-flag indicates that the RREQ needs special handling. The group leader, after receiving such a RREQ, grants the node permission to rebuild the tree by unicasting a RREP back to the node. It notes that it has given this node rebuilding permission and must not grant any other node such permission unless the current rebuild fails. Again, this is to prevent multiple nodes from attempting repairs (which would likely cause the formation of loops).

After receiving a RREP granting it rebuilding permission, the node unicasts a RREQ to the other group leader, using the node from which it received the Group Hello as the next hop. This RREQ contains the current value of the partition’s multicast group sequence number. When it receives the RREQ, the other group leader notes the set R-flag, takes the larger of its record of the group's sequence number and the received sequence number for the group, and increments this value by one. It then unicasts a RREP back to the source node. This group leader becomes the leader of the reconnected tree. As the RREP travels back to the source, it grafts a branch on to the tree. Having noted the R-flag, the next time the group leader sends a Group Hello, it sets the U-flag. All members formerly contained in the other partition (including the partition's group leader) note the new group leader information, and the merging of the two trees is then complete.

2.4.2 ODMRP: On-demand Multicast Routing Protocol

ODMRP (Lee, Gerla et Chiang, 1999) applies *on-demand* routing techniques to avoid channel overhead and improve scalability. It uses the concept of *forwarding group (Chiang, Gerla et Zhang, 1998a)*, a set of nodes responsible for forwarding multicast data on shortest paths between any member pairs, to build a forwarding *mesh* for each multicast group. By maintaining and using a mesh instead of a tree, the drawbacks of multicast trees in mobile wireless networks (e.g., intermittent connectivity, traffic concentration, frequent tree reconfiguration, non-shortest path in a shared tree, etc.) are avoided. A *soft-state* approach is taken in ODMRP to maintain multicast group members. No explicit control message is required to leave the group.

Multicast Route and Membership Maintenance

In ODMRP, group membership and multicast routes are established and updated by the source *on demand*. Similar to on-demand unicast routing protocols, a request phase and a reply phase comprise the protocol (see Figure 2.4). While a multicast source has packets to send, it periodically broadcasts to the entire network a member advertising packet, called a JOIN REQUEST. This periodic transmission refreshes the membership information and updates the route as follows. When a node receives a non-duplicate JOIN REQUEST, it stores the upstream node ID (i.e., backward learning) and rebroadcasts the packet. When the JOIN REQUEST packet reaches a multicast destination, the destination creates or updates the source entry in its Member Table. While valid entries exist in the Member Table, JOIN TABLES are broadcasted periodically to the neighbors. When a node receives a JOIN TABLE, it checks if the next node ID of one of the entries matches its own ID. If it does, the node realizes that it is on the path to the source and thus is part of the forwarding group. It then sets the FG -flag and broadcasts its own JOIN TABLE built upon matched entries. The JOIN TABLE is thus propagated by each forwarding group member until it reaches the multicast source via the shortest path. This process constructs (or updates) the routes from sources to destinations and builds a mesh of nodes, the *forwarding group*.

The forwarding group concept is visualized in Figure 2.5. The forwarding group is a set of nodes in charge of forwarding multicast packets. It supports shortest paths between any member pairs. All nodes inside the "bubble" (multicast members and forwarding group nodes) forward multicast data packets. Note that a multicast destination can also be a forwarding group node if it is on the path between a multicast source and another destination. The mesh provides richer connectivity among multicast members compared to trees. Flooding redundancy among forwarding group helps overcome node displacements and channel fading. Hence, unlike trees, frequent reconfigurations are not required.

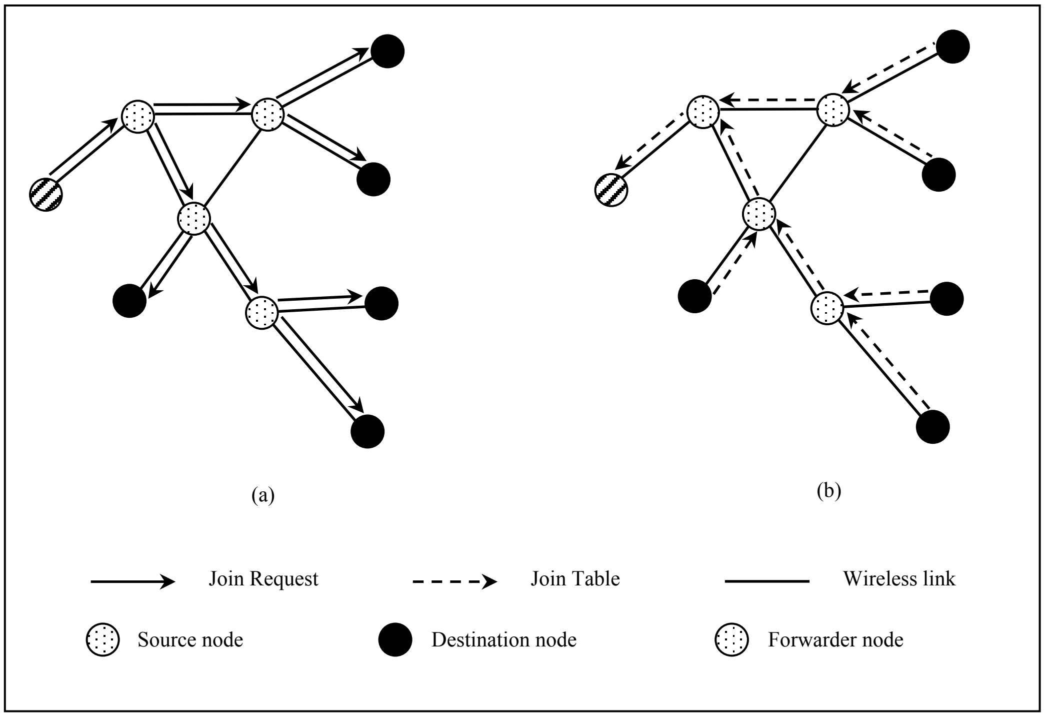


Figure 2.4 On-Demand Procedure for Membership Setup and Maintenance.
Taken from Lee (1999, p. 1299)

Figure 2.6 is an example to show the robustness of a mesh configuration. Three sources (S_1, S_2, and S_3) send multicast data packets to three destinations (R_1, R_2, and R_3) via three forwarding group nodes (A, B, and C). Suppose the route from S_1 to R_2 is S_1-A-B-R_2. In a tree configuration, if the link between nodes A and B breaks or fails, R_2 cannot receive any packets from S_1 until the tree is reconfigured. ODMRP, on the other hand, already has a redundant route (e.g., S_1-A-C-B-R_2) to deliver packets without going through the broken link between nodes A and B.

Figure 2.7 is shown as an example of a JOIN TABLE forwarding process. Nodes S_1 and S_2 are multicast sources, and nodes R_1, R_2, and R_3 are multicast receivers. Nodes R_2, and R_3 send their JOIN TABLES to both S_1 and S_2 via B, and R_1 sends its packet to S_1 via A and to S_2 via B. When destinations send their JOIN TABLES to next hop nodes, an intermediate node A sets the FG-flag and builds its own JOIN TABLE since there is a next node ID entry in the JOIN TABLE received from R_1 that matches its ID. Note that the JOIN TABLE built by A

has an entry for source S_1 but not for S_2 because the next node ID for S_2 in the received JOIN TABLE is not *A*. In the meantime, node *B* sets the FG-flag, constructs its own JOIN TABLE and sends it to its neighbors. Note that even though *B* receives three JOIN TABLES from the receivers, it broadcasts the JOIN TABLE only once because the second and third table arrivals carry no new source information. Channel overhead is thus reduced dramatically in cases where numerous multicast receivers share the same links to the source.

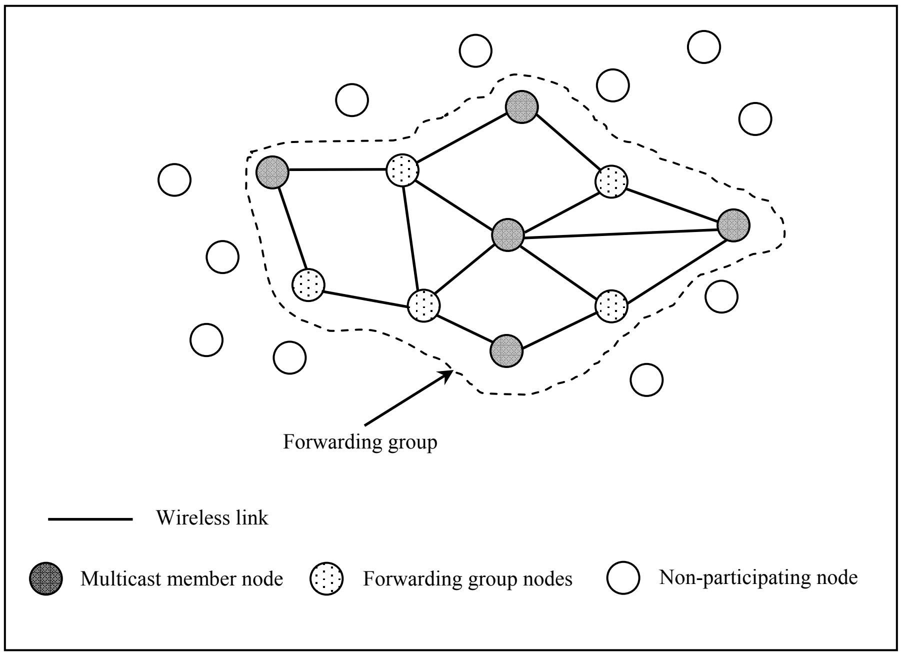


Figure 2.5 The Forwarding Group Concept.
Taken from Lee (1999, p. 1299)

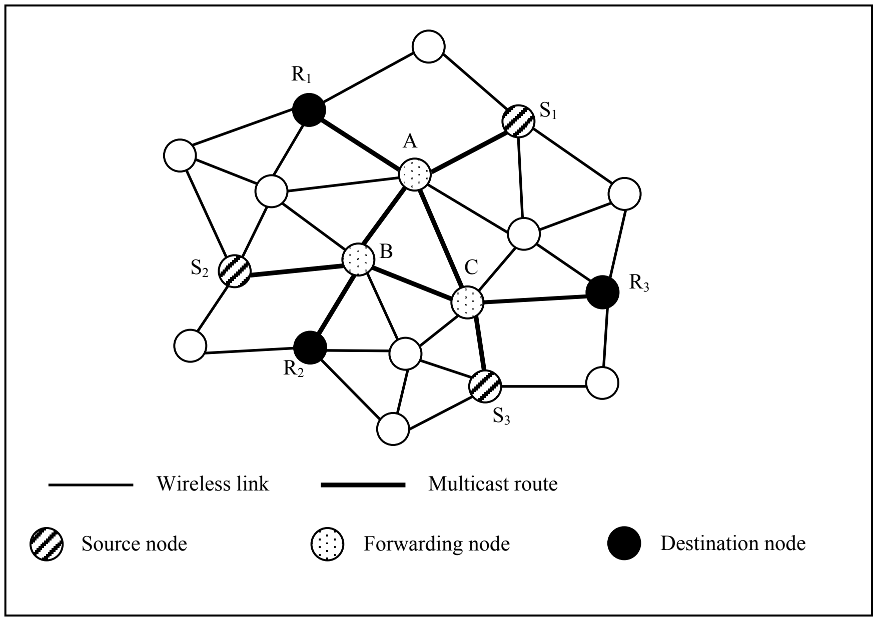
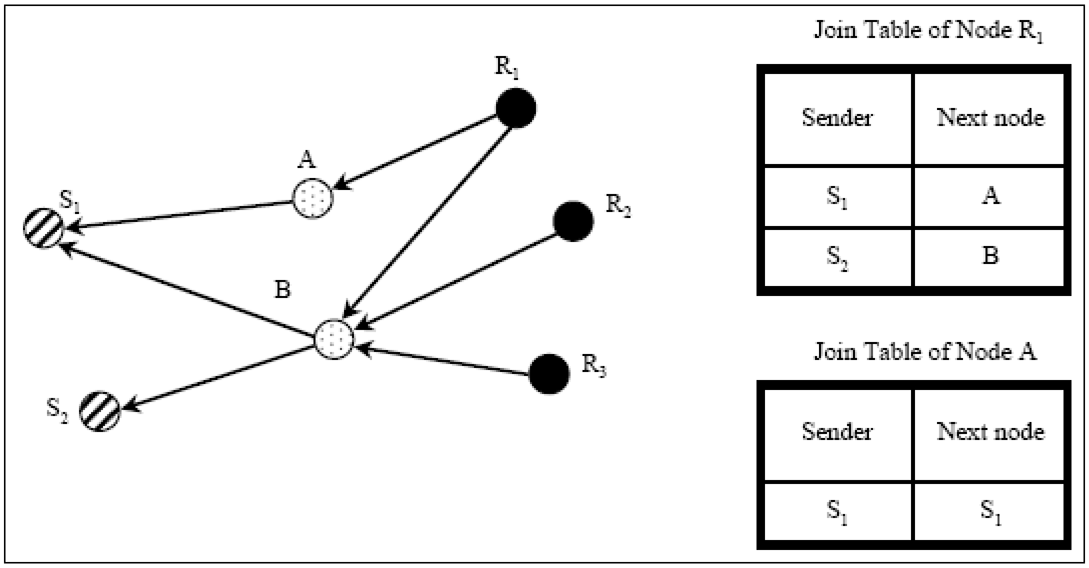


Figure 2.6 Why mesh?
Taken from Lee (1999, p. 1299)



Join Table of Node R_1

Sender	Next node
S_1	A
S_2	B

Join Table of Node A

Sender	Next node
S_1	S_1

Figure 2.7 An Example of a Join Table Forwarding.
Taken from Lee (1999, p. 1299)

Data Forwarding

After the group establishment and route construction process, a multicast source can transmit packets to receivers via selected routes and forwarding groups. Periodic control packets are sent only when outgoing data packets are still present.

When receiving a multicast data packet, a node forwards it only if it is not a duplicate and the setting of the FG-flag for the multicast group has not expired. This procedure minimizes traffic overhead and prevents sending packets through stale routes.

Soft State

In ODMRP, no explicit control packets need to be sent to join or leave the group. If a multicast source wants to leave the group, it simply stops sending JOIN REQUEST packets since it does not have any multicast data to send to the group. If a receiver no longer wants to receive from a particular multicast group, it removes the corresponding entries from its Member Table and does not transmit the JOIN TABLE for that group. Nodes in the forwarding group are demoted to non-forwarding nodes if not refreshed (no JOIN TABLES received) before they timeout.

2.4.3 ADMR: Adaptive Demand-Driven Multicast Routing Protocol

ADMR (Jetcheva et Johnson, 2001) supports the traditional IP multicast service model of allowing receivers to receive multicast packets sent by any sender (Deering, 1989), as well as the newer source-specific multicast service model in which receivers may join a multicast group only for specific senders (Holbrook et Cain, 2006). As in both multicast service models, a node need not be a receiver for the group to be able to send to the group, senders need not declare their intention to send multicast packets to the group before doing so, and senders need not explicitly declare their intention to stop being multicast senders. In addition, multicast sources do not need to know who the receivers are and receivers do not need to know who the sources are.

Multicast State Setup

Multicast state setup for a multicast group and source in ADMR is initiated by both the sources and the receivers for the group as each such node begins to send packets to the group or joins the group respectively.

When the routing layer at a node *S* receives a multicast packet for a group *G* from the application layer, if *S* is not currently a source for *G*, ADMR attaches an ADMR header to the packet and floods the packet in the network. When a node *R* who is a receiver for *G* receives this packet, it unicasts a RECEIVER JOIN packet back to *S*. As this packet is forwarded towards *S*, it sets up forwarding state for *S* and *G* in each node through which it is forwarded.

When the application on a node *R* first issues a join request for group *G*, ADMR floods a MULTICAST SOLICITATION packet in the network. Each source for *G* responds to the solicitation by unicasting a UNICAST KEEPALIVE packet back to *R*, which then responds with a RECEIVER JOIN to each source, which sets up forwarded state as described above.

The multicast forwarding state for a given multicast group *G* and sender *S* in ADMR forms a source-based multicast forwarding mesh. In some cases the nodes which are part of the source-based mesh represent a tree rooted at *S*, while in others, as a result of multicast state maintenance, additional nodes may become multicast forwarders, and thus the resulting set of nodes with forwarding state may no longer be a tree. In general, ADMR does not aim to create redundant state; such state may result as a side-effect of its normal operation.

Even when the set of nodes with multicast forwarding state may represent a source-rooted tree, packets in ADMR are not constrained to follow any particular branches or parent/child links but instead are *flooded* through all nodes with multicast forwarding state; such a node forwards each received packet regardless of which node transmitted the packet to it. Each packet is thus dynamically forwarded from *S* to the multicast receivers along the shortest-delay paths within the mesh; this set of end-to-end paths constitutes a tree rooted at *S*. For example, in Figure 2.8, packet *X* reaches receiver R_1 through node *B*, whereas packet *Y* reaches receiver R_1 through node *D*. This can happen when node *D* acquires the wireless medium before *B* and forwards the packet first, or when *B* does not receive the packet correctly due to wireless interference and is therefore unable to forward it.

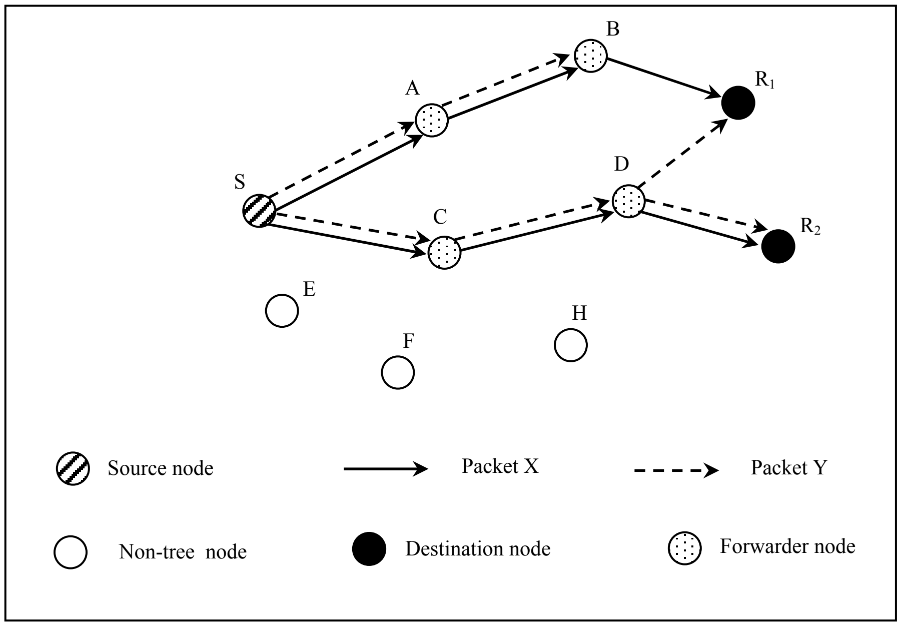


Figure 2.8 Multicast data packet forwarding within the ADMR mesh. Each packet traverses a source-rooted tree within the multicast mesh.
Taken from Jetcheva (2001, p. 35)

The flood of a packet constrained to the nodes with multicast forwarding state is known as a *mesh flood*, and the more general type of flood of a packet to all nodes in the network is known as a *network flood* (Figure 2.9). This use of flooding within the multicast forwarding mesh is similar to the “forwarding group” concept introduced in the FGMP protocol (Chiang, Gerla et Zhang, 1998a) and also used in ODMRP (Lee, Su et Gerla, 2002), except that in ADMR forwarding state is specific to each sender rather than being shared by the entire group. As a result, when a sender using ADMR sends a multicast packet, it floods within the multicast forwarding mesh *only* towards the group's receivers, whereas when using FGMP or ODMRP, the packet also floods back towards any other senders even though they may not be multicast receivers for the group. Although the source-specific forwarding requires ADMR to maintain source-specific state in forwarding nodes, such state is required anyway in order to support the source-specific multicast service model (Holbrook et Cain, 2006). In addition, even FGMP and ODMRP require source-specific state at each node, since they must detect duplicate packets during a flood within the forwarding group, and any type of packet identifiers used for this duplicate detection when there may be multiple group senders must be source-specific.

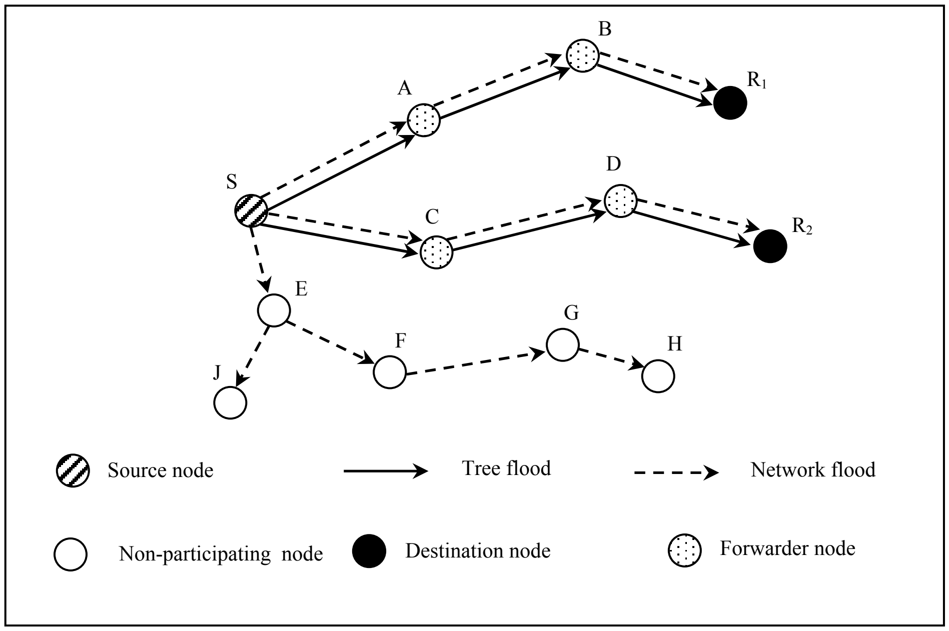


Figure 2.9 Mesh flood vs. Network flood.
Taken from Jetcheva (2001, p. 35)

Multicast Packet Forwarding

Any packet with a broadcast destination address containing an ADMR header will be sent as a network flood, i.e., it will be flooded to all nodes in the network. Any packet with a multicast destination address containing an ADMR header will only be sent as a mesh flood, i.e., it will be flooded to all nodes with multicast forwarding state for the source and group indicated in the packet. For either type of flood, the identification number in the packet's ADMR header is recorded in the Node Table of each node that forwards it; this information is used for duplicate detection so that any node that should forward the packet as part of a flood would forward no more than one copy of it.

When a node receives a packet from node *S* with a multicast destination address *G* and an ADMR header in it, it checks its Membership Table entry for *S* and *G* to determine if it should forward the packet. Whether or not it forwards the packet, the receiving node compares the hop count in the received packet's ADMR header to the hop count in this node's Node Table entry for *S*. If the new hop count is less than that already recorded in the Node Table entry, the node updates the entry with the new hop count and sets the previous hop address in the entry to the MAC-layer source address from which the packet was received. In addition, if the node forwards the packet, before doing so, it increments the hop count field in the packet's ADMR header and copies the packet's MAC-layer source address into the ADMR header previous hop address field.

Finally, if the packet has a payload following the ADMR header, the node checks its Membership Table to determine if it is a receiver member for *S* and *G*. If so, it passes the packet up within the protocol stack to allow the packet to be processed as a received multicast packet. Since each node in the multicast source mesh forwards each packet that is mesh flooded without regard to who transmitted the packet to it, each packet will flow within the mesh to the group receivers without being constrained to follow any specific links, and will thus be able to automatically be forwarded around temporarily broken links or failed forwarding nodes in the mesh.

New Multicast Source

When a node *S* originates a multicast packet for some group *G* for which it is not currently an active sender, it will not have a Sender Table entry for *G*. In this case, node *S* creates and initializes a new Sender Table entry for *G*. The expected packet inter-arrival time in this entry may be set to a default value, may be assumed based on the port numbers used in the packet, or may be specified by the sending application if an API is available for this purpose. Node *S* also inserts an ADMR header in the packet, includes a multicast group address option with the address of the multicast group in it, and sets the IP destination address of the packet to the network broadcast address, thus causing the packet to be sent as a network flood.

After flooding the first data packet, node *S* buffers for a short time (e.g., INITIAL-PKT BUFFER TIME) subsequent multicast packets that it might originate for group *G*, rather than sending them immediately as they are generated. This initial delay allows forwarding state to be set up and ensures that network resources will not be wasted if there are no receivers for *G* in the network.

Once *S* receives at least one RECEIVER JOIN packet from a receiver for the multicast group, and INITIAL PKT BUFFER TIME has elapsed, *S* sends all packets for *G* that are in its Send Buffer as normal multicast packets. Pacing techniques could optionally be applied in sending these buffered packets, in order to avoid congestion caused by sending them all at once as soon as allowed by the protocol. The packet exchange which takes place when a new source becomes active is depicted in Figure 2.10.

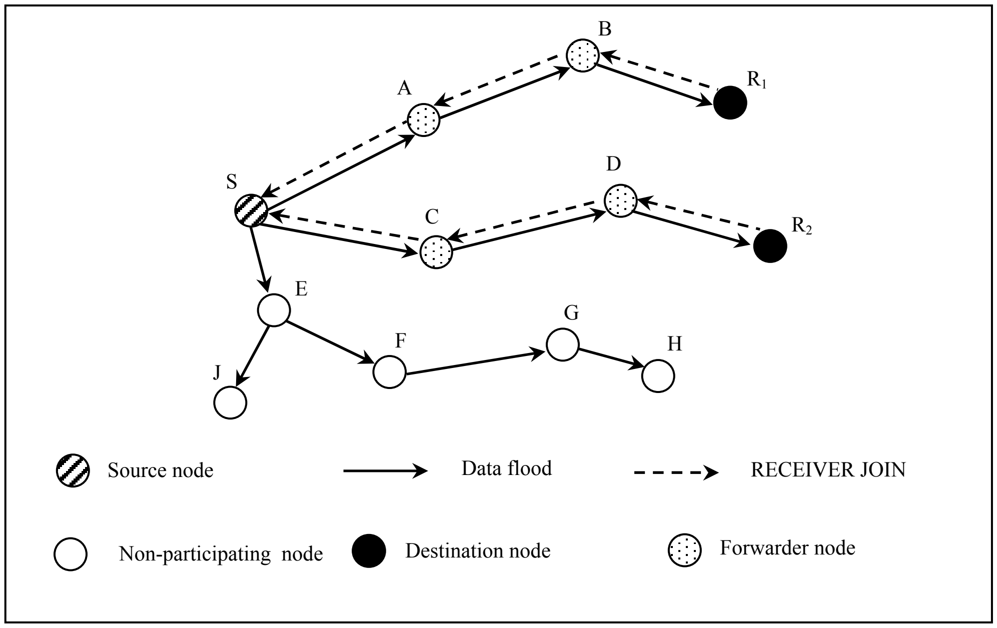


Figure 2.10 New multicast source. Node S sends a data flood to which multicast receivrs reply with RECEIVER JOIN.
Taken from Jetcheva (2001, p. 37)

Most subsequent multicast packets for *G* from *S* will be flooded within the multicast mesh for *S* and *G*, rather than being network flooded. However, it is possible that some interested receivers did not receive the initial data flood from *S* due to packet loss or a network partition at the time of the flood. To allow for such occurrences, node *S* uses a network flood rather than a mesh flood for certain of its subsequent multicast packets. The time between each packet selected to be sent as a network flood is increased until reaching a slow background rate, designed to tolerate factors such as intermittent wireless interference or temporary partition of the ad hoc network. These network floods are sent only when there is data to be sent to *G* and are delayed from their scheduled time until a new multicast data packet is originated from the application layer on *S*. The interval until each subsequent network flooded data packet is relative to the time at which the previous network flood data was sent, in order to preserve the approximate spacing between the floods.

Multicast Application Join

When an application on some node *R* requests to join a group *G*, the ADMR routing layer on node *R* a MULTICAST SOLICITATION packet as a network flood (Figure 2.11). This packet includes the group and source (for source-specific joins) of the group *R* is interested in joining and is intended to announce to sources for *G* that *R* is interested in receiving multicast data from them. If the group is a source-specific multicast group, the specific sender address *S* requested by the application is included as an additional option after the ADMR header.

The MULTICAST SOLICITATION is forwarded by all nodes in the network, except that in the case of source-specific multicast, the specified source *S* does not forward the MULTICAST SOLICITATION packet. Also in this case, if a node receiving the MULTICAST SOLICITATION has a Node Table entry for *S*, and has a Membership Table entry for *S* and *G*, indicating that it is a forwarder, this node will not rebroadcast the packet but will *unicast* it only to the previous hop address indicated in its Node Table entry for *S*; the node which receives this unicast packet will repeat the procedure and the packet will thus reach the multicast source through the multicast mesh (Figure 2.12). This mechanism speeds up the receiver join and potentially reduces overhead.

When any source *S* for multicast group *G* receives a MULTICAST SOLICITATION packet, it replies to the MULTICAST SOLICITATION to advertise to *R* its existence as a sender for the group. This reply may take one of two forms. If the next scheduled network flood of a multicast data packet is to occur soon, *S* may choose to advance the time for this network flood and use it as a reply to *R*'s solicitation. This form of reply is appropriate, for example, when many new receivers attempt to join the group at about the same time, since *S* would then receive a MULTICAST SOLICITATION from each of them, but could use the single existing network flood of the next data packet to reply to all of them. The other form that this reply may take is for *S* to send an ADMR keep-alive packet unicast to *R*. This keep-alive is a recently sent multicast data packet or a packet with an ADMR header and no payload and follows the reverse of the path taken by *R*'s MULTICAST SOLICITATION. Each node forwarding this unicast keep-alive packet unicasts it to the address recorded in the previous hop address field in its Node Table entry for *R*, created when the node forwarded *R*'s MULTICAST SOLICITATION flood (Figure 2.13). When forwarding the unicast keep-alive packet toward *R*, each node updates its Node Table entry for *S* in the same way as it would for a flood from *S*, recording the node on the path back to *S* in the entry's previous hop address field.

If node *S* replies to the MULTICAST SOLICITATION from *R* by sending a unicast keep-alive, as described above, then *S* also sets a timer and expects to receive a RECEIVER JOIN from *R* within a short time. If *S* does not receive the RECEIVER JOIN (Figure 2.14) within some time, then the unicast keep-alive was probably lost before reaching *R*, and so *S* will retransmit it. If the timer expires a second time and *S* has not received a RECEIVER JOIN from *R*, then *S* assumes that the path that the unicast keep-alive is trying to traverse is broken, and advances its next scheduled network flood of a multicast data packet.

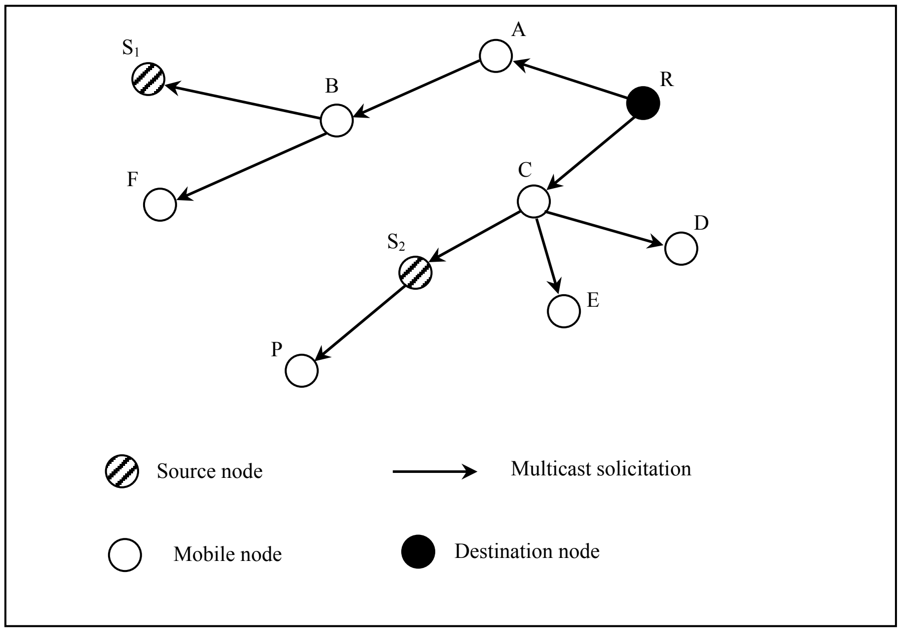


Figure 2.11 Receiver R floods a MULTICAST SOLICITATION.
Taken from Jetcheva (2001, p. 37)

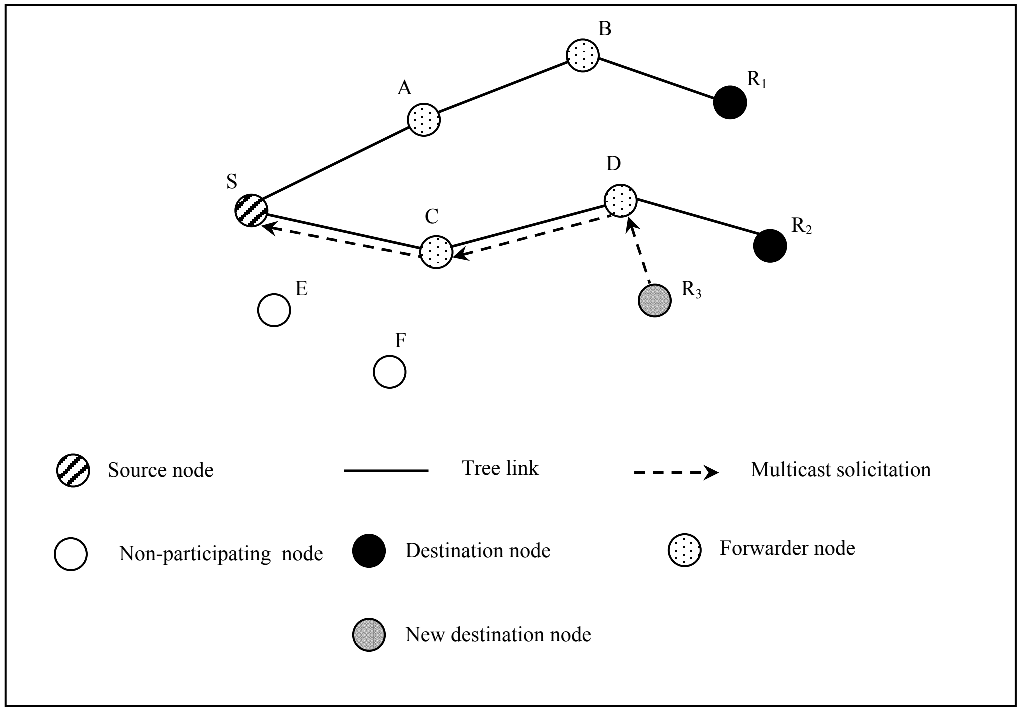


Figure 2.12 Node D receives a source-specific MULTICAST SOLICITATION and unicasts it on to the source.

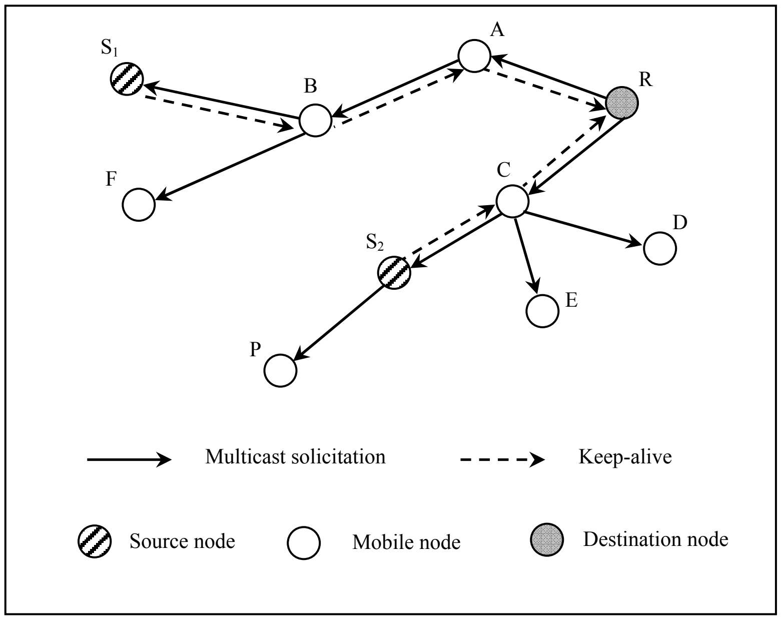


Figure 2.13 Sources S1 and S2 respond to receiver R's MULTICAST SOLICITATION.
Taken from Jetcheva (2001, p. 37)

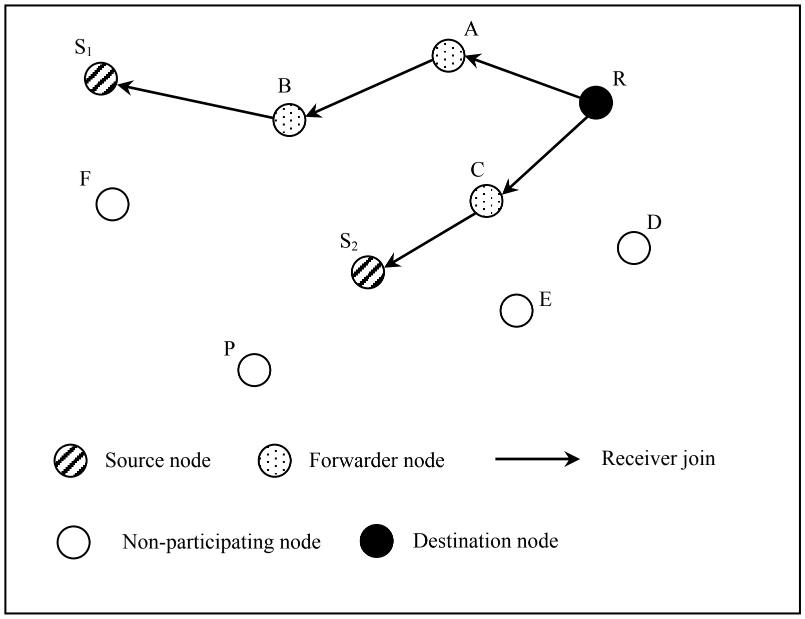


Figure 2.14 Receiver R unicasts a RECEIVER JOIN to each source from which it received a keep-alive.
Taken from Jetcheva (2001, p. 37)

Multicast State Maintenance

Forwarders or receiver members for source *S* and group *G* may become disconnected from the multicast forwarding mesh for *S* and *G* as nodes in the network move or as wireless propagation conditions change. ADMR performs state maintenance for each source's multicast forwarding mesh by monitoring the flow of traffic from the source to the multicast receivers, detecting when as a result of a broken link there is no path from the source to a receiver(s). Once such a link is discovered, ADMR performs repair procedures in order to restore the flow of data to the multicast receivers.

ADMR tracks the inter-arrival time of multicast packets sent by the application on node *S* to group *G* and computes an expected inter-arrival time for the application's packets. The expected packet inter-arrival time may be set to a default value, may be assumed based on the port numbers used in the packet, or may be specified by the sending application if an API is available for this purpose.

If the application layer at node *S* originates no new multicast packets for *G* within some multiple (e.g., 1.5) of this current expected packet inter-arrival time, the routing layer at S begins originating "keep-alive" packets for *G*; a keep-alive packet is a recently transmitted data packet or a packet with an ADMR header and no payload. These keep-alives are multicast to *G* just as any regular data packet and are used to maintain the existing forwarding state for the multicast forwarding mesh for *S* and *G*. They are only sent for a limited period of time after which the multicast application is assumed inactive and all multicast state for *S* and *G* in the network is expired.

Each forwarder or receiver for source *S* and group *G* detects that it has become disconnected from the multicast forwarding mesh when it fails to receive a number of successive expected multicast data (or keep-alive) packets (e.g., 3) from *S* for *G*.

Each node maintains a *disconnection timer* for each group and source for which it is either a forwarder or a receiver member, and resets this timer each time it receives a data or keep-alive packet for *G*. The timer value is based on the expected packet inter-arrival time value in the ADMR header of the last received packet, plus a time proportional to the node's hop count from the source *S*, as recorded in the last packet received from *S* (which updated the node's Node Table entry for *S*). This small increase in disconnection timeout value as a function of hop count is intended to generally allow the node directly downstream of the broken link to detect that the link is broken before nodes further downstream from *S* react to the lack of data and keep-alives and initiate repair procedures. Even if multiple nodes initiate repair procedure for a link that is not adjacent to all of them, the protocol would still operate correctly, but would generate more overhead packets than necessary to reconnect the forwarding mesh.

Local Repair

Although each multicast packet is forwarded within the forwarding mesh without regard to how previous packets were forwarded, the paths actually taken by each packet form a tree rooted at the multicast source. The tree "traced" by the last forwarded multicast packet is the one that ADMR will attempt to repair. Repairing this tree ensures connectivity between the multicast source and the receivers; attempting to restore the forwarding mesh is not necessary, and would generate unnecessary overhead.

When some node *C* detects disconnection from the forwarding mesh for source *S* and group *G*, it initiates local repair. This repair starts by node *C* attempting to ascertain that it is indeed disconnected from the source, and by also notifying nodes downstream from it (i.e., closer to the receivers) that it is attempting to reconnect the mesh, so that no redundant local repair attempts would take place. In particular, node *C* sends a REPAIR NOTIFICATION packet, which is forwarded towards the multicast receivers along the paths followed by the last forwarded multicast data or keep-alive packet (Figure 2.15). This packet is intended to reach all nodes that received the last forwarded multicast packet through *C* and who will thus possibly detect the disconnection themselves soon. These nodes will detect disconnection later than *C* does, since their disconnection timeouts are computed based on their higher hop counts from *S*.

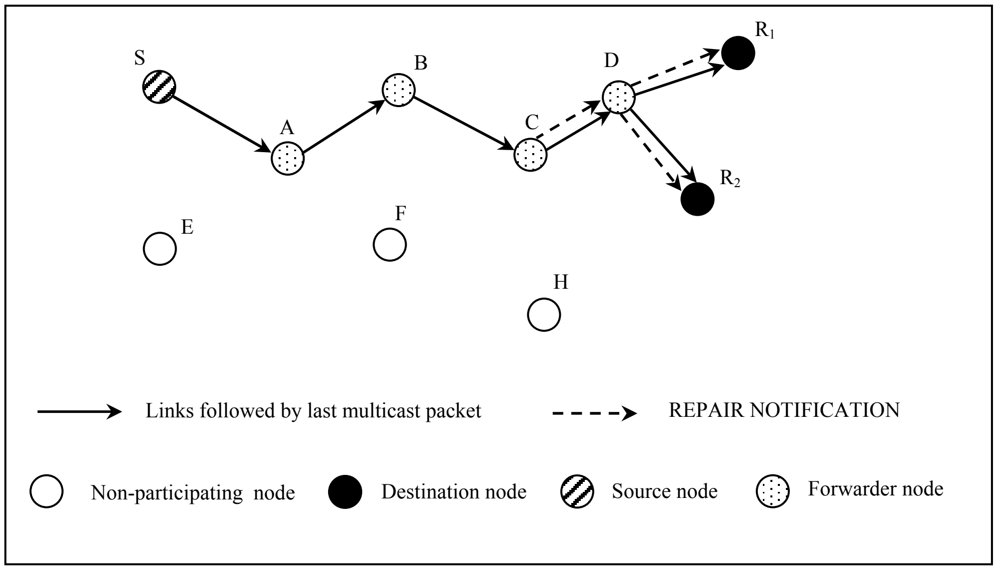


Figure 2.15 Node C downstream of break initiates local repair.
Taken from Jetcheva (2001, p. 38)

To forward the REPAIR NOTIFICATION packet to the nodes in the sub-tree below *C*, each node processes the packet only if the MAC-layer transmitting source address of the packet matches the previous hop address stored in that node's Node Table entry for the multicast sender *S*. After sending the REPAIR NOTIFICATION packet, node *C* waits for a START REPAIR DELAY period of time before proceeding with its local repair. If, during this delay, node *C* receives a REPAIR NOTIFICATION initiated by an upstream node for this same group and source, then *C* cancels its own local repair, since the repair should be performed by the node that is adjacent to the broken link, and *C* clearly is not.

The REPAIR NOTIFICATION packet serves two purposes. It is a notification to nodes in the sub-tree below C that a local repair is in progress and that they should not initiate their own local repair. It is also a chance to double-check that the link to node *C*'s parent node is indeed the one that is broken. The REPAIR NOTIFICATION will be received by nodes directly below *C* in the forwarding tree traced by the last forwarded multicast packet, and if the link from *C* to its parent B in the tree is actually not broken, may also be received by *B*. In the REPAIR NOTIFICATION packet, *C* lists the address of the node that is currently its parent, as represented by the previous hop address in its Node Table entry for *S*. If the REPAIR NOTIFICATION is received by this parent node, it will recognize that one of the nodes directly below it in the tree (node *C*) is performing a local repair and will send a one-hop REPAIR NOTIFICATION to *C*, causing it to cancel its local repair as described above.

When a receiver member for *G* receives (or sends) a REPAIR NOTIFICATION, it postpones its disconnection timer for a LOCAL REPAIR DURATION interval of time, which is an estimate of the amount of time the local repair is expected to take. If this timer expires and the receiver has not started to receive keep-alives or data packets, it will re-join the group as described later.

After START REPAIR DELAY time has elapsed, if node *C* has not received a REPAIR NOTIFICATION initiated by an upstream node for *S* and *G*, it will send a hop-limited RECONNECT packet as a limited network flood (Figure 2.16). The hop limit for the

RECONNECT packet (e.g., 3) limits this flood to only reaching nodes near *C*. The RECONNECT packet also includes the hop count from *S* to *C* recorded in *C*'s node table entry, so that the RECONNECT can be processed specially by multicast forwarding nodes upstream of the broken link, which are connected to the part of the mesh still connected to *S*. A forwarding node for *S* and *G*, e.g., node *F*, which has a smaller hop count to S than the one contained in the RECONNECT packet assumes that it is upstream of the repair node *C* and that it is therefore part of the mesh still connected to source *S*. Rather than forwarding the RECONNECT packet as part of the hop-limited network flood, node F reinitializes the packet's hop count limit to the default value, e.g., 255, and unicasts the packet to the node listed as its parent in the previous hop address field in its Node Table entry for *S* and *G* (node *A* in Figure 2.16). This copy of the RECONNECT is no longer treated as a network flooded packet, but instead is forwarded by each node that receives it to its parent in the mesh in the same way, until reaching *S*. If node *F* is in fact not upstream from the repair node *C* the unicast RECONNECT will reach *C*, which will discard it. Instead, if node *F* is upstream of *C*, the RECONNECT would reach *S,* which will respond with a RECONNECT REPLY packet. This RECONNECT REPLY packet is unicast back to the repair node *C* along the path traversed by the RECONNECT, recorded in the Node Table entries for *C* in the nodes along that path (Figure 2.17). Each node that forwards the RECONNECT REPLY packet joins the mesh for *S* and *G*, recording that it is a forwarder for *S* and *G* in its Membership Table.

The RECONNECT packet is forwarded all the way to the multicast source in order to ensure proper reconnection. Due to packet loss and node movement, it is not possible to always have accurate hop count information at all forwarding nodes. This may lead a disconnected node to reconnect to the multicast mesh through a node that is in the sub-tree below it and is therefore in the part of the forwarding mesh that is disconnected from the source.

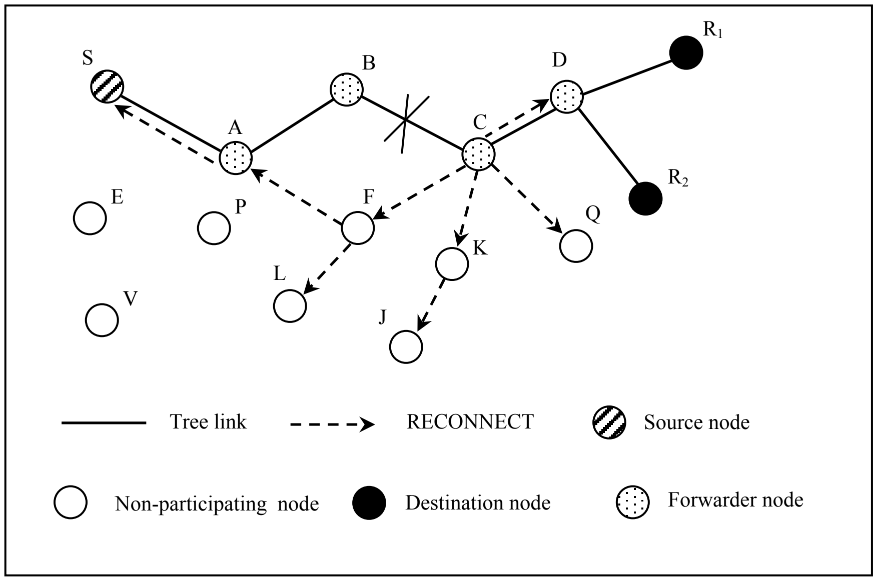


Figure 2.16 Disconnected Node C sends a RECONNECT packet.
Taken from Jetcheva (2001, p. 38)

Global Repair

If the local repair procedure described previously succeeds, the multicast forwarding mesh will be reconnected and the receiver members will start to receive multicast packets. If LOCAL REPAIR DURATION time elapses and the receivers have not received any multicast packets from *S* for *G*, this is an indication that the local repair has probably failed, perhaps because the amount of mobility in the network has been too high to allow the type of hop-limited repair attempted. In this case, each receiver performs its own individual repair by rejoining the group and source in the same way that it joins a group for the first time, as described previously.

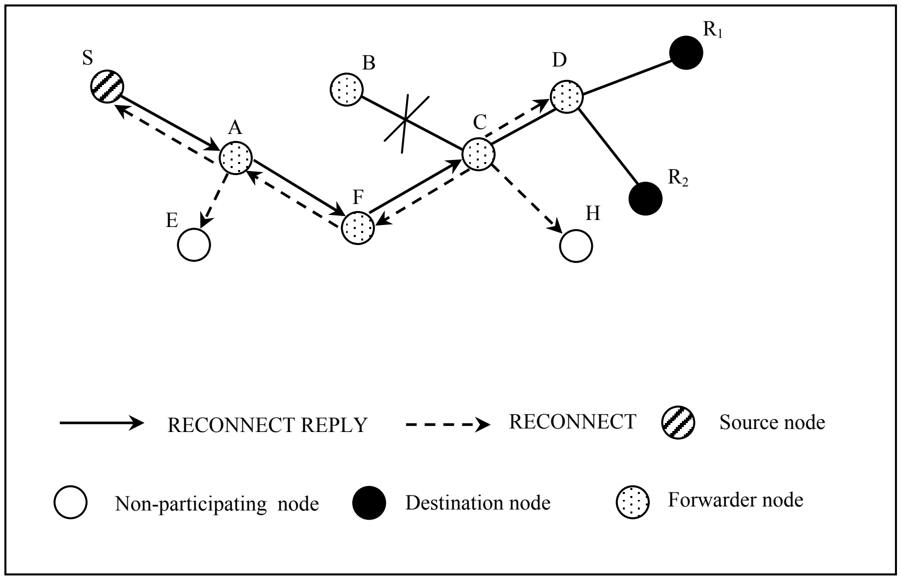


Figure 2.17 Source S sends a RECONNECT REPLY to node C.
Taken from Jetcheva (2001, p. 39)

2.5 Related Work on Video multicast Over Wireless Ad Hoc Networks

This section describes the existing researches related to video multicast routing over wireless ad hoc networks.

2.5.1 Multiple Tree Video Multicast over Wireless Ad Hoc Networks

Wei and Zakhor proposed multiple tree construction schemes and routing protocols for video multicast over wireless ad hoc networks (Wei et Zakhor, 2007). The basic idea is to split the video into multiple parts and send each part over a different tree, which are constructed to be disjointed with each other so as to increase robustness to loss and other transmission degradations. The two schemes are: Serial Multiple Disjoint Trees Multicast Routing (Serial MDTMR), and Parallel Multiple Nearly-Disjoint Tree Multicast Routing (Parallel MNTMR).

Serial MDTMR constructs two node-disjoint trees in a distributed way. Similar to ODMRP (On-Demand Multicast Routing Protocol) (Lee, Su et Gerla, 2002), group membership and multicast trees in Serial MDTMR are established and updated by the source on demand. At first, serial MDTMR build a shortest path multicast tree. Then after requiring all the middle nodes in the first tree not to be middle nodes of the second tree, it constructs another shortest path tree. Since these two trees do not share middle nodes at all, they are node disjoint. When a multicast source has packets to send, it periodically triggers a two-step multicast tree construction/refresh process. In the first step, the multicast source broadcasts to the entire network a *JoinRequest* message, which includes the treeID. When a node receives a non-duplicate *JoinRequest* message for the first tree, it stores the upstream node ID, and rebroadcasts the message. When the *JoinRequest* message reaches a multicast destination, the destination unicasts a *JoinAck* message to the multicast source via the reverse shortest path. When a middle node in the reverse path receives a non-duplicate *JoinAck* message, it updates its corresponding forwarding state in the Forwarding Table, and forwards the message to its upstream node. Each middle node of the tree only forwards the *JoinAck* message once in one tree construction cycle.

After receiving the first *JoinAck* message, the multicast source waits for a short time period before broadcasting another round of *JoinRequest* message for the second tree in order to ensure the disjointness of two trees. When a node receives a non-duplicate *JoinRequest* message, it forwards the packet only if it is not a middle node of the first tree in this round. When the *JoinRequest* message reaches a destination, the destination unicasts back a *JoinAck* message to the multicast source to set up the second tree.

However, Serial MDTMR does not guarantee destination connectivity to both multicast trees. Moreover, it suffers from a tremendously high control overhead. To overcome these weaknesses, Parallel MNTMR protocol is proposed. This protocol constructs two nearly-disjoint multicast trees in a single routine. Parallel MNTMR randomly classifies all the nodes in the network into one of two categories, i.e., group *x* and group *y*. The protocol could potentially build tree-*x* purely from nodes in group-*x*, and tree-*y* purely from nodes in group-*y*, resulting in two node-disjoint trees. Figure 2.18 shows how multicast trees are constructed in Serial MDTMR protocol.

Parallel MNTMR has two control packets to construct two trees with both high tree connectivity and low tree similarity. The two packets are: Join-Query (JQ) and Join-Reply (JR) packets. Each node, say *z*, in group *x* should perform the following condition before it decides to forward any control packets. The *sorting condition*: a JQ message received by a node *z* satisfies the *sorting condition*, either if it is the first JQ message that node *z* receives in the current JQ round, or if the following two conditions are satisfied: (a) the number of hops it has travelled is no larger than that of the first received JQ message of node *z* plus one and (b) the JQ message has not been forwarded by node *z*. The *forwarding condition*: is satisfied if the following two conditions hold true: (a) node *z* has not forwarded a JQ message in this JQ round, and (b) the message's last hop is the sender or a group-*x* node. The *upstream node condition*: if a node *z* has received JQ messages of the both groups, group-*x* and group-*y*, it selects last hops of the earliest received group-*x* and group-*y* JQ messages as upstream nodes for tree-*x* and tree-*y*, respectively. Otherwise, if node *z* has received only JQ messages of group-*y*, node *z* selects last hops of the earliest and the second earliest received JQ messages as upstream nodes for tree-*x* and tree-*y*, respectively; otherwise if node *z* has received one JQ message, the last hop of the only JQ message is selected as upstream nodes for both tree *x* and *y*.

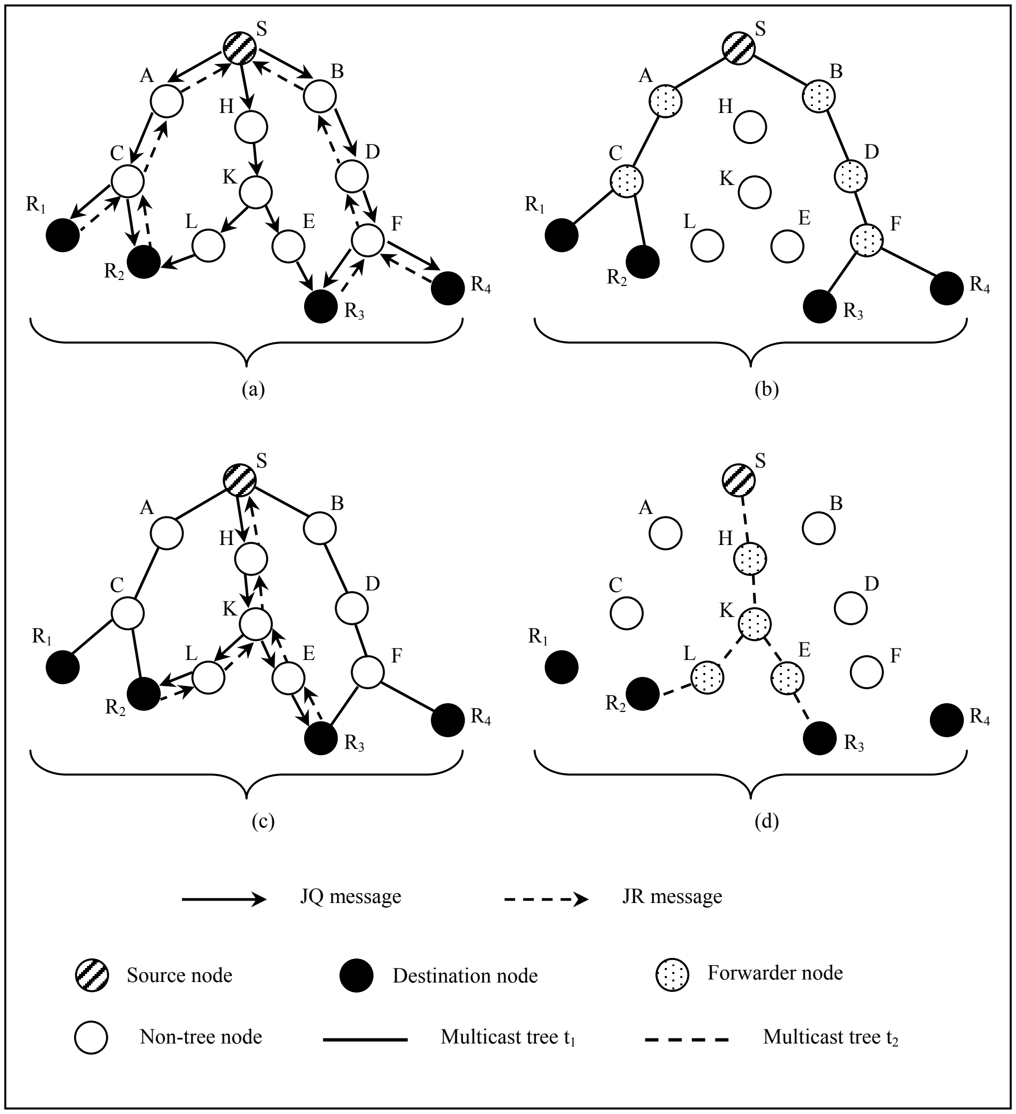


Figure 2.18 Multicast trees construction in Serial MDTMR:
(a) First round JQ and JR messages. (b) First multicast tree.
(c) Second round JQ and JR messages. (d) Second multicast tree.

In parallel MNTMR, Tree construction is performed in a similar way as in ODMRP. When a multicast source has packets to send, it triggers a multicast tree construction process by broadcasting a JQ message to its neighbors. When a node receives a group-*y* JQ message, if the message satisfies the *storing condition*, the node stores it into the *JQ message cache* for later usage in the JR process; otherwise, the message is simply discarded. If the message also satisfies the *forwarding condition*, the current node forwards the JQ message to its neighbors immediately; otherwise, if the JQ message is the earliest received JQ message in the current

JQ round, the node sets a JQ-delay timer. When the JQ-delay timer expires, if the node has not forwarded a JQ message in this JQ round, it forwards the earliest received JQ message at that time. The JQ-delay scheme tends to make *pure JQ messages* be selected and forwarded with a priority over *mixed JQ messages* in the distributed tree construction process.

When a destination receives a group-*y* JQ message, if the message is a *pure JQ message*, and the node has not initiated a JR message in this JQ round for tree-*y*, it selects the last hop of this JQ message as its upstream node for tree-*y*, and unicasts a JR message to the sender via the selected upstream node, in order to set up tree-*y*. All nodes, receiving and forwarding the JR message for tree-*y*, become middle nodes for tree-*y*. The destination also sets a timer upon receiving the earliest JQ message. When the timer expires, for each tree for which it has not already initiated a JR message, the destination selects an upstream node according to the *upstream node selection rule* and unicasts a JR message to the sender via the selected upstream node to construct that tree.

When a middle node receives a non-duplicate JR message for tree-*x*, it selects an upstream node according to the *upstream node selection rule*, and forwards the JR message to the upstream node. In the end, two trees are obtained. The first tree, mainly, consisting of group-*x* nodes and the second tree mainly consisting of group-*y* nodes. Therefore, these two trees are likely to be nearly disjoint.

Parallel MNTMR achieves low tree connectivity when the node density is not high enough, since it is equivalent to partitioning the network into two networks with half of the node density. Route query messages may sometimes even be blocked in the middle of the network, causing some destinations, which are physically connected to the source, to be connected to neither tree. Both Serial MDTMR and parallel MNTMR protocols apply the periodic update scheme of ODMRP to maintain the tree structure. However, these protocols usually require high density of forwarding nodes to ensure good connectivity because the multicast group members do not participate in data forwarding. This requirement becomes more stringent in parallel MNTMR because the topology is virtually divided into two parts. In addition, the multicast trees are constructed at the beginning of the multicast session and a mechanism to allow a node to join a multicast session at anytime is not presented.

The tree construction process is visualized in Figure 2.19. The network consists of one source (S), two destinations (R_1 and R_2), and five other nodes. The dashed lines denote the underlying topology of the network, dot-dashed arrows denote the construction of the first tree, and solid arrows denote the construction of the second tree. We assume that nodes A,B, and E belong to group-1, and nodes C and D belong to group-2. According to Parallel MNTMR, both R_1 and R_2 select node E as their upstream node for tree t_1, and node D as their upstream node for tree t_2. Let JQ_E and JQ_D denote node sets of last hops of JQ messages stored in JQ messages caches of nodes E and D, respectively. $JQ_E\{B,C\}$ and $JQ_D=\{A,C\}$. According to the *upstream node selection rule*, node E selects node B as the upstream node for tree t_1, since node B is a group-1 node, while node C is a group-2 node. Using the same rule, node D selects node C as the upstream node for tree t_2. Thus, two disjoint trees are constructed, where intermediate nodes of tree t_1 are nodes B and E, and those of tree t_2 are nodes C and D. Note that each node learns the group of its candidate upstream nodes through JQ messages.

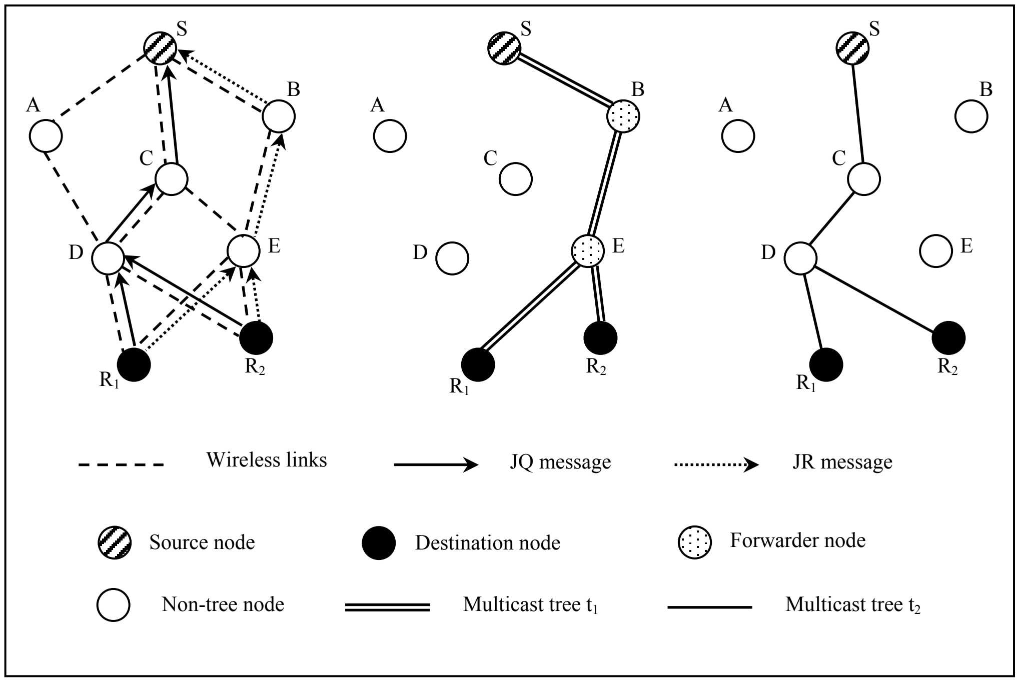


Figure 2.19 Multicast trees construction in Parallel MDTMR.
Taken from Wei (2007, p. 8)

2.5.2 Robust Demand-Driven Video Multicast Routing over Ad hoc Wireless Networks

Agrawal et al. presented a multiple tree protocol called Robust Demand-driven Video Multicast Routing (RDVMR) . RDVMR builds based on Adaptive Demand Driven Multicast Routing (ADMR) protocol (Jetcheva et Johnson, 2001). ADMR achieves very low normalized packet overhead and high packet delivery ratio compared to other conventional multicasting protocols like ODMRP, MZRP (Zhang et Jacob, 2004), and MAODV (Royer et Perkins, 2000). ADMR is a receiver-initiated multicasting protocol, and it uses hard- state tree repair. Its salient features are: (i) There is no periodic control traffic like beaconing, or link state updates. (ii) Most of its control information is piggybacked on data packets. (iii) It is standalone, i.e., it does not require any underlying unicast routing protocol.

RDVMR explores the path diversity and error resilience properties of MDC. RDVMR deploys a novel path based Steiner tree heuristic to reduce the number of forwarding nodes in each tree, and constructs multiple trees in parallel with a reduced number of common nodes among them to provide robustness against path breaks and to reduces the total data overhead.

RDVMR builds and maintains K ($K = 2$) trees. RDVMR is a tree-based receiver-initiated multicasting protocol, where destinations join and leave on-demand. RDVMR adapts each tree to the continuously changing network topology and to the changing group membership. RDVMR has three routing packets responsible for building and maintaining the multiple trees. They are *MulticastSolicitation*, *ScrJoinAdvt*, and *LocalReconnect* packets originating by a destination, source, or a forwarder for a particular group, respectively. In order to maintain K trees, each packet contains the *treeId*, identifying the tree it is meant for, in addition to the *groupId*, identifying the multicast group it is meant for.

The source of the group (*groupId*) multicasts data packets along the tree. It also periodically floods a data packet having a piggybacked header called *SrcJoinAdvt*, which eventually reaches every node in the network. It advertises a route to the source of *groupId* to all nodes in the network, and helps to refresh the multicast tree. Since the *SrcJoinAdvt* packet advertises a route to the source along all trees, it has its *treeList* set to all ones, whereas the packets multicasted by the source advertise a route to it only along a particular tree *t*. On hearing such a route advertisement to the source of the group *groupId*, a node creates (or refreshes) the entry *TreeState* (it contains the parent node for the tree *t*, multicast *groupId*, the last sequence number, and the cost to reach the source node). This way every node keeps track of its parent node for the tree *t* to reach the source. The source also keeps track of the mean inter-packet time *ipt*, which is piggybacked on every packet it sends. A tree node (i.e., a node that is a forwarder or a destination) is said to be disconnected if it has not received any multicast packet within a small multiple of *ipt*. In order to keep the multicast tree intact, the source also multicasts *keep alive* packets every *ipt* if it does not have any data packets to send. *Keep alive* packets are multicasted along the tree just like data packets.

When a destination node wishes to join the tree *t*, it unicasts a *joinReq* packet to its parent for the tree *t*. In case it does not have a parent, it floods a *MulticastSolicitation* packet for the tree *t*. When a tree node (i.e., a node that is a forwarder or a destination for tree *t*) gets a *MulticastSolicitation*, it unicasts it up the tree through its parent to the source. On receiving a *MulticastSolicitation*, the source unicasts a *joinReply* packet to the corresponding destination. A node forwarding a *joinReply* packet to the node *x* for the tree *t*, becomes a forwarder for that tree.

If a forwarder for tree *t* detects a disconnection, it multicasts a *repairNotification* packet downstream. A *repairNotification* serves to inform the downstream nodes of the disconnection, and avoids redundant repair attempts by them when they eventually detect the disconnection. Simultaneously the forwarder detecting the break, attempts to locally repair the tree by flooding a limited TTL *LocalReconnect* for the tree *t*. Similar to handling a *MulticastSolicitation*, a tree node unicasts a *LocalReconnect* up the tree *t* to the source. The source then unicasts a *LocalReconnectReply* to the node that originated the local repair. A *LocalReconnectReply* is processed exactly like a *joinReply*, i.e., any node forwarding it becomes a forwarder for the tree *t*. In order to avoid routing loops, only the source is responsible to respond to flooded packets like *MulticastSolicitation* and *LocalReconnect*. If a destination detects a disconnection, it attempts to rejoin the group after some time by flooding a *MulticastSolicitation* in case the repair attempts of its upstream nodes (i.e., nodes that are ancestors of this node along the multicast tree) fail.

RDVMR does not have explicit *leave* messages, instead forwarders prune themselves away if they detect that they do not have any downstream node. Each multicasted data packet carries a field containing the parent of the node that forwarded it. This field serves to passively acknowledge the parent of that node to continue forwarding data. However, since pure destinations do not forward data packets, they need to explicitly acknowledge their parents by periodically unicasting an acknowledgment packet. This way a forwarder not having any downstream destinations prunes itself away from the multicasting tree.

RDVMR deploys a novel path-based Steiner tree heuristic, which tries to reduce the number of additional forwarding nodes required by differentially costing each node along each tree.

By differentially costing destinations, forwarders, and non-tree nodes, and by encouraging links to serve more downstream destinations, the number of forwarders nodes is reduced.

The basic motivations behind the proposed cost function are: (i) *Increasing the number of downstream receivers under each link*: By assigning a lesser cost to a forwarder with more number of downstream children under it, the multicast efficiency (the bushyness of the tree) increases. (ii) *Making as many receivers as forwarders*: By assigning a lesser cost to destinations, the number of extra forwarding nodes needed per tree is reduced. (iii) *Decreasing the number of forwarders*: A higher cost is assigned to non-tree nodes, hence promoting paths to the source which have the lowest branch cost to the tree. (iv) *Increasing* the node disjointedness of the $(K = 2)$ trees: In order to reduce the number of shared nodes among different trees, the cost also includes a disjointedness component, which is higher for nodes being forwarders for multiple trees, and zero for nodes which are a part of just one tree. Note that this conflicts with reducing the number of forwarders, and a tradeoff between them is achieved by the parameters γ and λ in the cost function. The parameters γ and λ control the tradeoff between the node disjointedness and the number of forwarders.

The cost of a node, say *x*, can be calculated as follows. The cost is initialized as:

$$baseCost = NON\_PART\_OF\_TREE\_COST$$

Then, if node *x* is a destination for tree *t* its cost will be:

$$baseCost \times \alpha$$

Otherwise, if node *x* is a forwarder for tree *t* it calculates its cost as follows:

$$(baseCost \times \beta)/ncr$$

where *ncr* is the number of destinations descendents in the sub-tree rooted at node *x* for the tree *t*. α and β are less than one. α is set to be much lower than β to have a destination a much lower cost than a forwarder.

Finally, if the number of trees of node *x* (*numTx*), in which node *x* is a forwarder node is larger than one, the cost of node *x* is then:

$$\gamma \times (disjointedCost) + (1-\gamma) \times baseCost$$

where the *disjointedCost* equal to:

$$disjointedCost = \lambda \times NON\_PART\_OF\_TREE\_COST \times (numTx - 1)$$

The cost function is illustrated in Figure 2.20. In this example, Source S floods a join request having a piggybacked cost. Node R_6 is the new receiver. The join requests arrive at node R_6 through four paths, of which the one through A and B is chosen as it has the least cost. In this example, *NON_PART_OF_TREE_COST* = 100, $\alpha = 0.03, \beta = 0.8$, and $\gamma = 0$

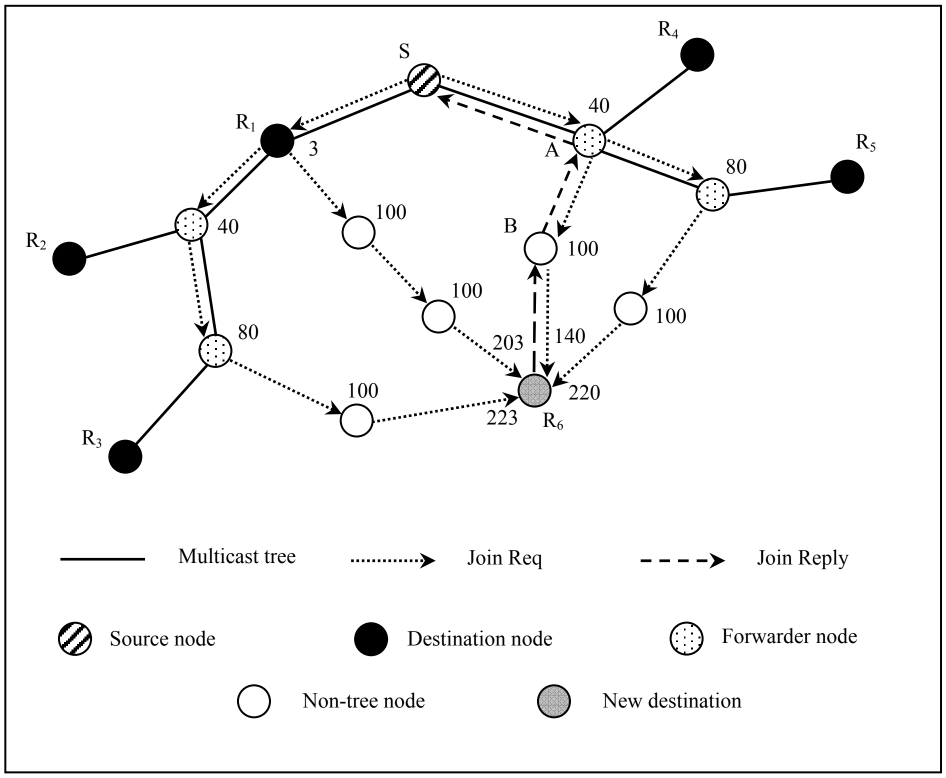


Figure 2.20 Illustrative example of the cost function.
Tirée d'Agrawal (2006a, p. 7)

2.5.3 Multiple Tree Multicast Ad Hoc On-Demand Distance Vector (MT-MAODV) Routing Protocol

MT-MAODV, based on MAODV, constructs two optimally disjoint multicast trees in a single routine for video multicast (Chow et Ishii, 2008). MDC scheme is used to split the video into several independent and equally important video descriptions. Each description is transmitted over different tree.

Multicast trees construction in MT-MAODV is performed, as in MAODV, by exchanging three control packets: RREQ_J (join request), RREP_J (join reply), and MACT_J (multicast activation). MT-MAODV adds a filed in the control packets of MAODV to construct multiple multicast trees. The field indicates the status of each mobile node. The status of a node can be any of multicast group member (ON_GROUP), forwarding node for both tree-1 and tree-2 (ON_TREE_0), forwarding node of tree-1 (ON_TREE_1), forwarding node of tree-2 (ON_TREE_2), and not tree member (NOT_ON_TREE). A node wants to join a multicast group, it changes its status to ON_GROUP if it is a tree member of both multicast trees (ON_TREE_0). Otherwise, it needs to send a (RREQ_J) packet. A node can only unicast the RREQ_J if it is the first time this node initiates a join request and the unicast route is recorded in the Group Leader (GL) table. Each intermediate node is allowed to forward only one RREQ_J and the immediate neighbor of the requesting node must record its ID in the *first_hop* field of the RREQ_J packet. RREQ_J is replied by multicast tree members. A multicast group member is allowed to reply to at most two RREQ_J that arrive to it provided that the routes traversed by these RREQ_J are completely disjoint. To check the route disjointedness, the *first_hop* of these RREQ_J are inspected. Since each intermediate node is allowed to forward only one RREQ J, it can be easily realized that routes having different *first_hop* are fully disjoint. Besides, the *tree* field of the RREQ_J is copied to the *req_tree* field of the RREP_J to indicate the priority of this RREP_J; for example, a RREP_J(*tree* = 1, *req_tree* = 2) is low priority because it is desired to join tree-2 but the reply is given by member of tree-1. However, this type of RREP_J is delayed by a small interval to allow more suitable RREP_J to be sent first. The selection of best possible upstream node (RREP_J) is based on the following parameters in the sequence of decreasing priority: sequence number, hops to tree and hops to GL.

MT-MAODV maintains the connectivity of the multicast trees as follows. For every GROUP HELLO INTERVAL, GL of a multicast group broadcasts a group hello message (GRPH) to check the connectivity of each tree member and to update the group members with the latest information about the multicast group. Upon receiving a new GRPH, a node, regardless of its status, updates its GL table to reflect the currently active multicast group and the GL. If this node is a tree member, its MR table is also updated if this GRPH is received from its upstream node and the originator of this GRPH is the recorded GL in the MR table. In MT-MAODV, tree partition only occurs when both multicast trees are parted because a node can declare itself as GL only when it fails to connect to both multicast trees. Since group members are connected to both multicast trees, and so only a group member that fulfills the requirements as in MAODV is allowed to initiate a tree merge. A tree member that is not a group member can inform its nearest group member in the direction toward GL_1 by using GRPH with repair flag (GRPH_R). The RREQ_JR is broadcast to *GL*2 in a control manner. When an intermediate node receives a RREQ JR, it checks its GL table for the recorded GL for this multicast group. The RREQ_JR is re-broadcast if the recorded GL is GL_2; otherwise, it is discarded silently. Besides, if this node is an immediate neighbor of the requesting node, its ID is recorded in the *first_hop* field of the RREQ_JR. Upon receiving a RREQ_JR by GL_2, the information is cached and a waiting timer is initiated. Upon the timeout of this waiting timer, two next hops are selected and two RREP_JR that carry different values in their *tree* field are sent to the requesting node. Tree maintenance also involves tree repair, which is initiated by a downstream node that detects a link breakage. The same procedure as multiple tree construction, which involves the exchange of RREQ_J, RREP_J and MACT_J, is followed. The *tree* field is set accordingly to indicate which tree is to be repaired.

An example of multiple trees construction using MT-MAODV is shown in Figure 2.21. Assuming node *GL* is the group leader of the multicast group, nodes R_1, R_2, R_3 and R_4 subsequently join the multicast group one after one. Also, assume no information is recorded in their *GL* table. When node R_1 initiates a join request, two possible upstream nodes are *Y* and *Z*, and both options are replied by node *GL*; therefore, the *tree* field in both RREP_J are zero. Node R_1 can arbitrarily select one of them to be the upstream node (node *Y*) to tree t_1 and the other one (node *Z*) to tree t_2. Next, node R_2 initiates its join request and again, two replies are obtained; one from node R_1 (*tree* = 0) and the other one from node *Z* (*tree* = 2). Obviously, R_2 must select node *Z* as its upstream node to tree t_2 and node R_1 to tree t_1. Subsequently, the join request initiated by node R_3 returns three RREP_J: two from node R_1 (*tree* = 0) via nodes *X* and *W*, and one from node R_2 (*tree* = 0) via node *V*. Here, priority is given to connect to different tree or group members; therefore, either *X* or *W* is selected for one tree (for example, tree t_1), and node *V* is used for the other tree (tree t_2). Finally, node R_4 has only one option, which is node *U*. Since the priority is tree connectivity instead of tree disjointedness, node *U* must become the upstream node to both tree t_1 and tree t_2.

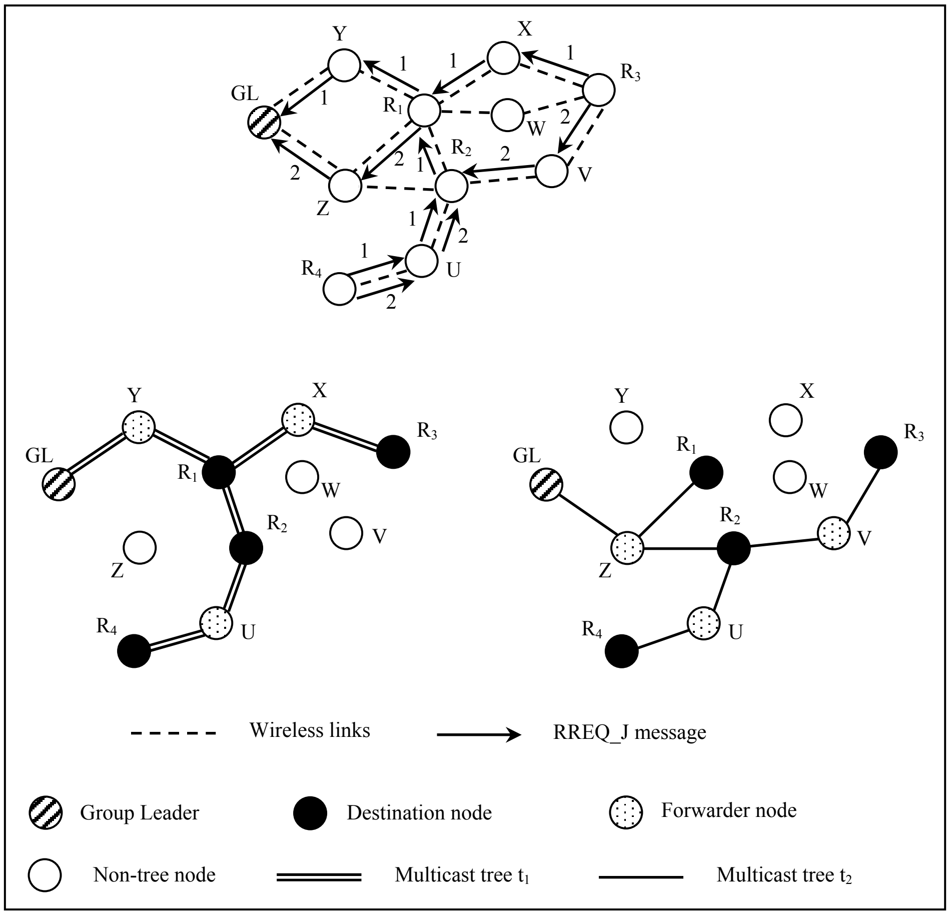


Figure 2.21 Multiple trees construction in MT-MAODV.
Taken from Chow (2008, p. 432)

2.5.4 Rate Adaptive Multicast

Rate-Adaptive Multicast (RAM) (Asif, Nguyen et Xu, 2006) routing algorithm is based on ODMRP (Lee, Su et Gerla, 2002). RAM, which uses a parameter referred to as the weight of the source-to-destination path. Given the optimal transmission rate *R* of a link (as determined by the signal strength of a received packet), the weight of the link is defined as 1/*R*. The weight of a path is the sum of the weights of all links on that path.

In the RAM protocol, JOIN-QUERY packets of ODMRP are used to estimate the channel conditions and to record the weights of the paths traversed by these packets. The weights are recorded in a field referred to as *pathWeight*. After receiving a JOIN-QUERY, every node on a path between the sender and receiver, measures the signal strength and suggests a rate, *SuggestedRate*, derived from the signal strength using an algorithm similar to RBAR (Holland, Vaidya et Bahl, 2001). The *SuggestedRate* is converted to the link weight, which is then added to the *pathWeight* recorded in the JOIN-QUERY.

A node participating in the route discovery/refresh process may receive multiple JOIN-QUERY packets from the same sender that have arrived from different paths. The node considers all of these JOIN-QUERY packets and selects the path with the minimum *pathWeight* value. If the node is a receiver, the selected path is the source-to-destination path with the lowest total transmission time among those considered. The receiver then creates a JOINREPLY that records the routing information and sends the JOIN-REPLY to the source, as in the ODMRP algorithm. To prevent the implosion of the JOIN-REPLY packets, every node on paths from the receivers to the sender consolidates information from several JOIN-REPLY packets it receives in the downstream direction (i.e., from a sender towards a receiver) into one JOIN REPLY, as in ODMRP. If a node is on the selected path, it uses the *SuggestedRate* recorded earlier for transmitting data packets at the physical layer.

Figure 2.22 shows the main components of our multicasting scheme for real-time transmission of video in MANETs (Mobile Ad Hoc Network). A Layer Coding (LC) technique known as scalable, non-causal prediction with vector quantization and conditional replenishment (SNP/VQR) is used to encode the raw video (Asif et Kouras, 2006). SNP/VQR offers layer coding where the video is encoded into multiple layers of data streams. A subset of the compressed bit stream provides the base quality. Additional enhancement layers build on to the video quality in both the spatial and temporal domains. Receivers with powerful batteries and access to higher bandwidth channels can choose to receive all layers, thus acquiring video of the highest quality. Other receivers may opt for a reduced number of layers to save energy and network bandwidth. Similarly, when a router detects the onset of congestion or a drained battery, it can select to transmit only the most important layer(s) using priority packet dropping (Bajaj, Breslau et Shenker, 1998).

A SNP/VQR encoder at the server compresses the video sequence frame by frame. A multicast sender, referred to as the video packetizer, transforms the compressed frame into an array of data packets, which are transmitted over the MANET. Because of the scalable nature of the codec, each subscriber has an option to select how many layers it wants to receive. After the selected layers are received, the SNP/VQR decoder reconstructs the compressed video frames. In case of transmission errors, SNP/VQR decoder applies error concealment techniques to the damaged pixels.

Although the rate information is updated periodically, the recorded rates may become inaccurate before the next route refresh round due to the mobility of the nodes. Furthermore, RAM protocol was tested under low mobility (i.e., 1 *m/s*). However, in case of high mobility, it is not clear how RMA protocol will perform.

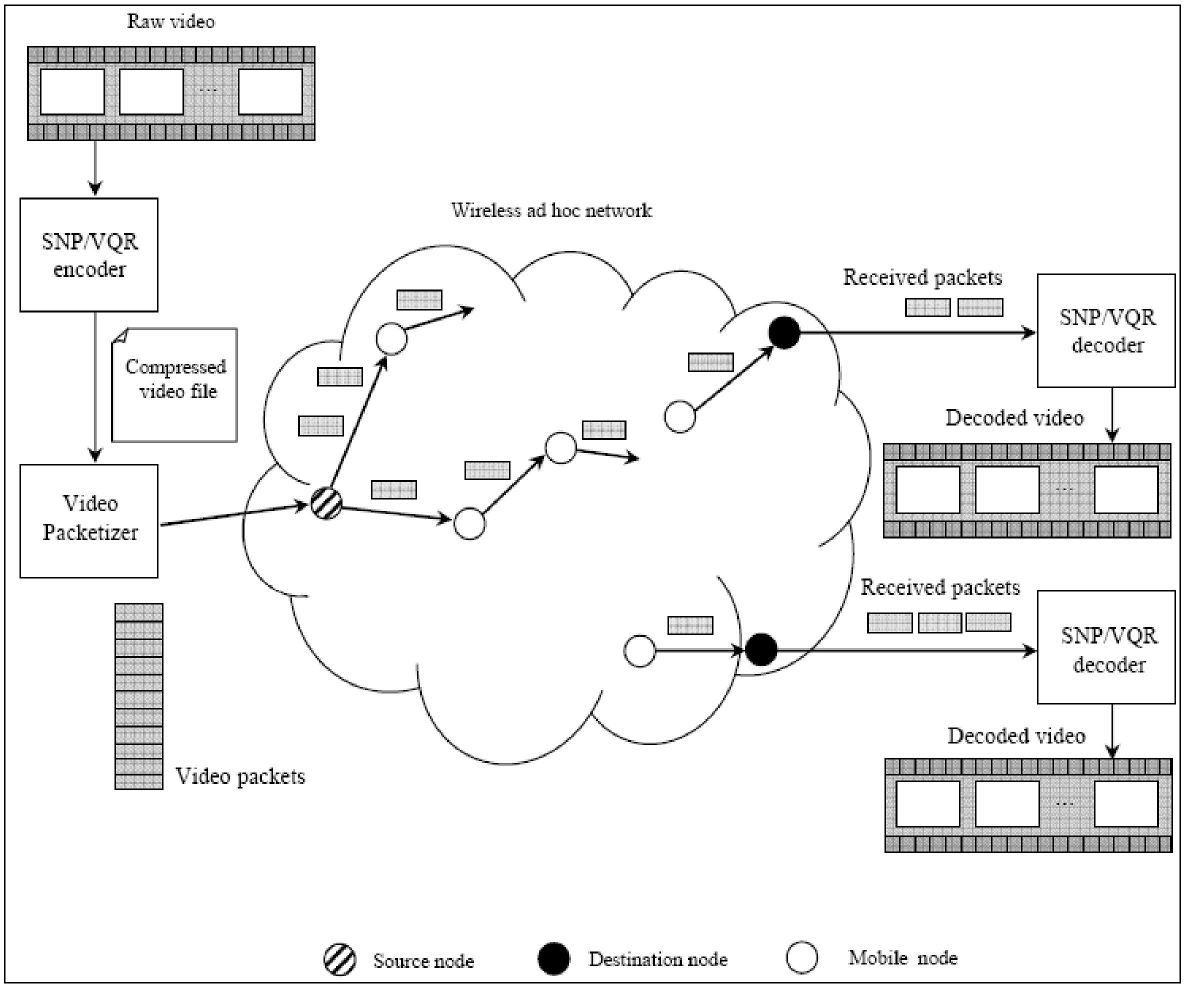


Figure 2.22 Components of the real-time video transmission framework for MANETs.
Tirée d'Asif (2006, p. 404)

2.5.5 Multi-tree Multi-layer Algorithm for MD Video Multicast

Authors in (Mao et al., 2006) proposed, an application-centric, cross-layer approach for MD video multicast. The main objective of this approach is to optimize the application layer performance metric, i.e., video distortion. They formulated the optimized multicast routing as a combinatorial optimization problem and proposed an efficient Genetic Algorithm (GA)-based metaheuristic solution procedure. In this approach, a number of multicast trees are used, with each multicast tree supporting one video description. Furthermore, each description is coded into a base layer and an enhancement layer in order to cope with diversity in wireless links bandwidth. Packets belonging to the same description from both the base and enhancement layers are transmitted on the same tree. Figure 2.23 illustrates the proposed multicast scheme for MD video. Since the video session is coded into two descriptions, two multicast trees are needed. Note that each video description is further coded into a base layer and an enhancement layer. In the example, the path from the source S to destination R_2 in Tree 1 has enough path bandwidth for both layers of Description 1. But in Tree 2, the path bandwidth between the source S and destination R_2 only has enough bandwidth for the base layer of Description 2. This MD video multicast approach can effectively deal with frequent link failures and diverse link qualities in wireless ad hoc networks. First, since a destination is a member of multiple trees, it is connected to the source node via multiple paths. When a path is broken due to a failed link, the other path(s) may still be in a good condition, assuming link failures in the network are independent. As a result, such path diversity provides enhanced error resilience to an MD video session (Apostolopoulos, 2001; Mao et al., 2003). Second, due to highly diverse wireless links, the end-to-end bandwidths from the source to the destination (path bandwidth) are also highly diverse.

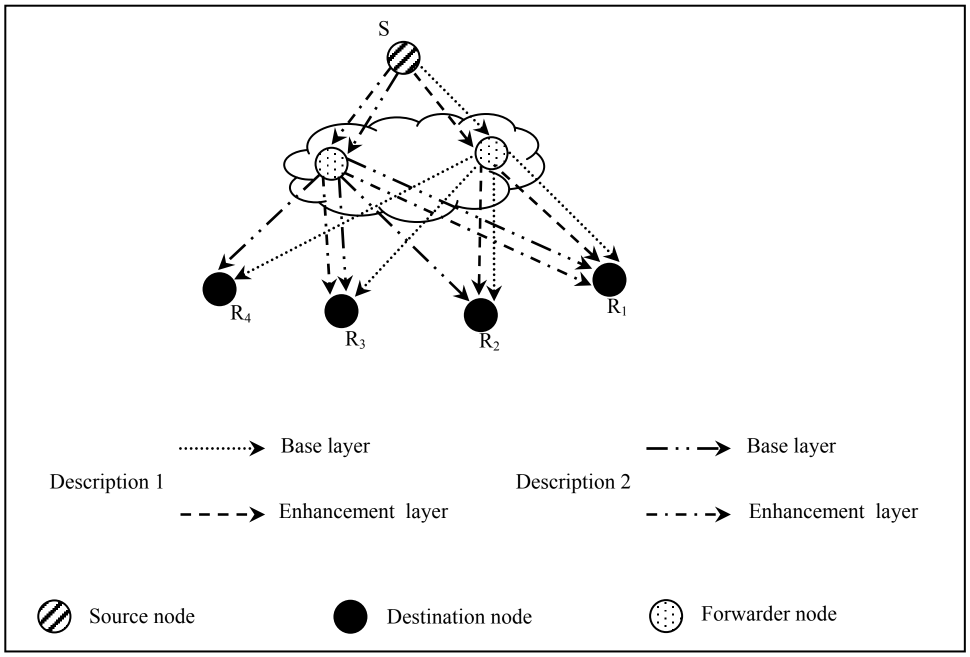


Figure 2.23 MD video multicasting using two trees.
Taken from Mao (2006, p. 64)

It is desirable to code each description into a number of layers, so that a destination with a high path bandwidth can subscribe the maximum number of layers of the corresponding description that are allowed by its path bandwidth for best video quality.

However, the authors focused on the network layer and they assumed the physical layer and MAC (Medium Access Control) layer dynamics are translated into network layer parameters. They focus on bandwidth, failure probability, and delay because these are the key characteristics of wireless links, as well as key factors that determine video distortion. The drawbacks of this approach are the dependence of a centralized computation and the complexity of the algorithm. However, nodes mobility, joining/leaving and rejoining a multicast group and multicast trees maintenance are not taken into consideration.

2.5.6 Optimized Video Multicasting over Wireless Ad Hoc Networks Using Distributed Algorithm

He et al. proposed a distributed algorithm to jointly optimize the source rate, the routing scheme and the power allocation using hierarchical dual decompositions (He, Lee et Guan, 2009). First, they applied a prioritized coding scheme to enable the heterogeneous receivers to reconstruct the video at the different quality levels. Then, they formulated the video multicast problem using the network model, the packet loss model, and the video distortion model.

The prioritized coding scheme consists of two steps. At the first step, each frame of the video sequence is encoded into N layers. The source bits in layer i ($i = 1, ..., N$) is denoted as R_i. At the second step, the source bits R_i is split into i equal source blocks, each containing bits R_i/i. Then (N - i) redundant blocks of 0s are padded to construct N blocks. In this way, N source packets, denoted from M^1 to M^N, are generated. The source packets, M^1, M^2, ..., M^N, are hierarchical. The decoding of a packet requires the existence of all the corresponding lower-layer packets. The hierarchical relationship can be removed with network coding. With random linear network coding (Ho et al., 2006), the source node sends out the encoded packets that are linear combinations of the original source packets, and the intermediate nodes forward the encoded packets that are linear combinations of previously received packets. The receiving nodes decode the received packets to get the original source information (Fragouli, Boudec et Widmer, 2006). The padding pattern of the 0s blocks is broadcasted to all the nodes before video transmission. If a receiver receives k encoded packets for a frame before the playback deadline, the receiver can recover the source bits from R_1 till R_k for the frame, thus reconstructing it at k-layer quality. It is not necessary for the receiver to receive all the N encoded packets in order to decode a frame. Even if some packets for a frame are lost, that frame can still be constructed at a partial quality.

2.6 Conclusions

In this chapter, we have presented the related works on video multicast over wireless ad hoc networks. The main objective of these works is to improve the quality of the received video by exploiting the error resilience properties of MDC along with multiple paths. In other words, MD video are encoded and transmitted over two different paths to each destination node. If only one path is broken, packets corresponding to the other description on the other path can still arrive at the destination node on time. Although there are algorithms (He, Lee et Guan, 2009; Mao et al., 2006) that take into consideration the heterogeneity of destination nodes, however, none of them attempts to increase the number of assigned MD video to the destination nodes. Instead, they aim at improving the quality of the received video. In addition, they do not consider node mobility, joining/leaving and rejoining a multicast group, and multicast trees maintenance. Furthermore, are their approaches feasible in wireless ad hoc networks? The answer is not clear.

However, all works on video multicast assigned MD video to the destination nodes in the same way, i.e., sequential assignment. This means that each destination nodes should be assigned the first description and the second description or it will be assigned the first description only. In other words, they do not consider the *independent-description* property of MDC while assigning MD video to destination nodes.

CHAPITRE 3

MULTICAST TREES CONSTRUCTION AND MULTIPLE DESCRIPTION ASSIGNMENT ALGORITHMS

3.1 Introduction

In this chapter, we present different algorithms for constructing multiple node-disjoint multicast trees and assigning MD video to a group of heterogeneous multicast destinations. The basic idea is to deploy the *independent-description* property of MDC along with multiple node-disjoint multicast trees to increase the number of assigned MD video to a group of destinations. We show that MDC can increase the number of assigned MD video when compared to LC.

Specifically, we propose three algorithms, namely, Serial MDC, Centralized MDC, and Distributed MDC. Serial MDC constructs multiple node-disjoint paths and assigns MD video description simultaneously. We refer to it as Serial MDC, because it constructs multiple node-disjoint paths and assigns MD video to each destination node one after another. Then after the construction of multiple node-disjoint paths and the assignment of MD are performed, Serial MDC constructs multiple node-disjoint multicast trees. In Centralized MDC, a multicast source node, first, constructs multiple node-disjoint multicast trees. It then assigns different video description to each multicast tree. Distributed MDC randomly assign MD video to the destination nodes in a distributed manner. Then, based on the assignment of MD video distributed MDC constructs multiple node-disjoint multicast trees. Furthermore, we propose MSPT and MSMT algorithms for multiple multicast trees construction. Both algorithms do not consider the *independent-description* property of MDC during the assignment of MD video. Instead, they assign them in a sequential manner.

The rest of this chapter is organized as follows. In section 3.2, we present our network model and problem formulation of video multicasting. In Sections 3.3, 3.4, 3.5, and 3.7, we describe our proposed algorithms for constructing multiple node-disjoint multicast trees and assigning

MD video. In Section 3.6, we evaluate our proposed algorithms. The complexity analysis of the protocols is presented in Section 3.8. Finally, our conclusions are presented in Section 3.9.

3.2 Network Model and Problem Formulation

3.2.1 Network Model for Multicasting

We consider a multi-hop wireless ad hoc network with the nodes V. The nodes communicate with each other via wireless links. Each node in the network can communicate directly with a subset of the other nodes in a network. A node v can transmit directly to node u if the both nodes are within the transmission range of each other. We modeled a wireless ad hoc network as weighted graph $G = (V, E)$, where V is a set of wireless nodes each with random location and E is a set of wireless communication links between the nodes. A link between node pair $\{v,u\}$ indicates that both nodes v and u are within each other's transmission range. The nodes in set V can be of the following three types:

- **Multicast source node:** The node that sends out the multicast video packets. We denote it by S.
- **Destination node:** A node that receives the multicast video packets. The set of destination nodes in a multicast tree is denoted by $\mathbf{Y} \subseteq V - S$.
- **Forwarder node:** A node that is an intermediate hop in the path from the source S to a destination node in $\mathbf{Y}$. It is denoted by F.

Two positive real-valued functions are defined on a link $e = \{v,u\} \in E$, namely:

- *Link Delay*: $d(e) \in \Re^{+}$.
- *Link Bandwidth*: $Bw(e) \in \Re^{+}$.

In this chapter, we focus on the network layer, i.e., the construction of multiple multicast trees and the assignment of MD video. We assume that the physical and MAC layers dynamics, such as the link delay and bandwidth, are translated into the network layer parameters. These parameters can be measured at every node and distributed through the network using LSAs (Link State Advertisements) (Clausen et Jacquet, 2003).

Definition 1: A path p from the multicast source S to a destination node in G is defined as a list of nodes $(v_1, v_2, \ldots, v_k)$ such that $\forall j, 1 \le j < k, e_j = \{v_j, v_{j+1}\} \in E$ and no node appears more than once.

The delay of the path p is the sum of all link delays, that is,

$$d(p) = \sum_{j=1}^{k-1} d(e_j) \qquad (1)$$

The bandwidth of the path p is the minimum available bandwidth of all links, which is defined as:

$$Bw(p) = \underset{e_j \in p}{min}\{Bw(e_j)\} \qquad (2)$$

In case of K node-disjoint paths, $P = \{p_1, p_2, \ldots, p_K\}$, then the delay of K paths for a destination node is:

$$d(P) = \underset{p_j \in P}{max}\{d(p_j)\} \qquad (3)$$

Let L be the number of the multicast trees constructed to meet the destinations' requirements, then the delay of tree-aggregate $T = t_1 \cup t_2 \ldots \cup t_L$ is defined as:

$$d(T) = \underset{l \in |1 \ldots L|}{max} d(t_l) \qquad (4)$$

where $d(t_l)$ is the delay of a multicast tree t_l, which is defined as the longest delay from the source S to the destinations on t_l, that is:

$$d(t_l) = \underset{p_j \in t_l}{max}\{d(p_j)\}, \quad j = |1 \ldots L| \qquad (5)$$

3.2.2 Problem Formulation

Our video multicast problem can be formulated as follows: given a network $G = (V, E)$ with n MD video, links delay, available links bandwidth, a source S, and set of destinations

$\mathbf{Y} = \{R_1, R_2, \ldots, R_m\}$. Each destination $R_i \in \mathbf{Y}$ requires a preference number of MD video, say $VD_{req}(R_i)$, then construct multiple node-disjoint multicast trees spanning $\mathbf{Y} \cup S$ such that the total number of the assigned video descriptions to each destination is maximized. That is:

$$maximize\{VD_{req}(R_i)\} \qquad (6)$$

To minimize the delay of every path from the source S to each destination $R_i \in \mathbf{Y}$, the shortest path tree algorithm is deployed. That is,

$$minimize\{d(p_{R_i})\}, \ \forall R_i \in \mathbf{Y} \qquad (7)$$

This problem is NP-complete in nature (Kompella, Pasquale et Polyzos, 1992) and a heuristic approach is used to solve the problem.

3.3 Serial Algorithm for MD Assignment

The MD video assignment and multiple multicast trees construction algorithms are shown in algorithms 1-4. At the beginning, let the multicast source has a partial topology that contains multiple paths to each destination, as shown in Figure 3.1(a). Following, it arranges the destinations that require one and two video descriptions in a descending order according to their number of node-disjoint paths in the sets *x* and *y*, respectively. After that, it checks the destinations in the set *y* if any of them has only one path, if yes, it adds it to the set *x*. At the end of these steps, the sets *x* and *y* contain the destinations arranged in a descending order according to their number of paths. After that, the source node runs the algorithms 1-4. We use the two colors: red and green to refer to the first and second descriptions, respectively. The multicast source starts with the set *y* and constructs its red (R) and green (G) paths for each destination if possible. To find the R-path, the green nodes (G-nodes) should be removed because they already have been assigned a description and they cannot be on another tree. However, to find the G-path, the red nodes (R-nodes) should be removed. The R and G paths are constructed using shortest path algorithm (in terms of delay).

When the set y is empty, the source node starts with the set x. Since *any description* can reproduce the original video signal, this, what we referred to as *independent-description* property of MDC, therefore the multicast source will assign any color (R or G) to each destination in the set x.

Based on the sets of multiple paths K_{R_i} (the R and G paths) for every destination R_i, then the multicast source S constructs multiple multicast trees for the video transmission according to algorithm 4. That is, all nodes that have been assigned the same color are attached to the same tree. For example, the nodes that have been assigned the R-color are attached to the first tree (R-tree) and the nodes that have been assigned the G-color are attached to the second tree (G-tree). Figure 3.1 is an illustrative example.

```
1.  **Given:** A partial topology $G = (V, E)$, *set x*, and *set y*
2.  **for** $\forall i \in$ *set y* **do**
3.      $Z = set\ y$
4.      Construct a $R_{path}$ using algorithm 2
5.      Construct a $G_{path}$ using algorithm 3
6.  **end for**
7.  **for** $\forall i \in$ *set x* **do**
8.      $Z = set\ x \cup set\ y$
9.      Construct a $R_{path}$ using algorithm 2
10.         **if** the $R_{path} = \phi$ **then**
11.             Construct a $G_{path}$ using algorithm 3
12.         **end if**
13. **end for**
```

Algorithm 1 MD Video Assignment.

```
1. **for** $\forall$ $i$-$G_{path} \in Z$ **do**
2.     $P$ = Parents of $G_{nodes}$
3.     $V \leftarrow V - P$
4. **end for**
5. Construct a $R_{path}$ using the shortest path (in terms of delay) algorithm
```

Algorithm 2 *R_{path}* Construction.

```
1. **for** $\forall$ $i$-$R_{path} \in Z$ **do**
2.     $P$ = Parents of $R_{nodes}$
3.     $V \leftarrow V - P$
4. **end for**
5. Construct a $G_{path}$ using the shortest path (in terms of delay) algorithm
```

Algorithm 3 G_{path} Construction.

```
1. **Given:** $Z = set\ x \cup set\ y$
2. **for** $\forall i \in Z$ **do**
3.     **if** $i$ has $R_{color}$ **then**
4.         Add $i$ to $R_{tree}$
5.     **else if** $i$ has $G_{color}$ **then**
6.         Add $i$ to $G_{tree}$
7.     **end if**
8. **end for**
```

Algorithm 4 Multiple Multicast Trees Construction.

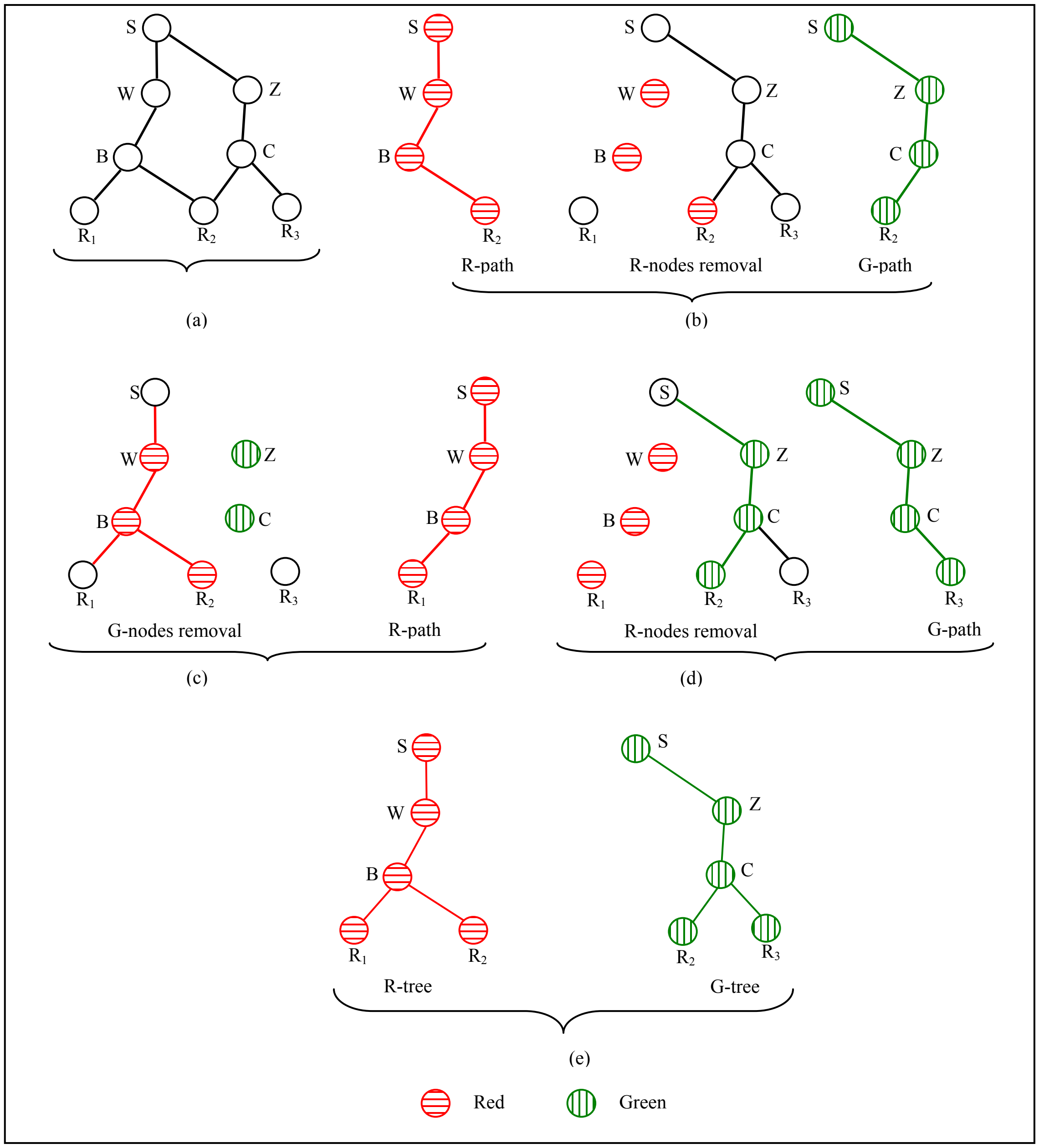


Figure 3.1 Demonstration of MD assignment and multicast trees construction. (a) Partial topology. (b) Multiple paths construction and nodes removal for destination R_2. (c) Multiple paths construction and nodes removal for destination R_1. (d) Multiple paths construction and nodes removal for destination R_3. (e) Multiple multicast trees construction.

3.4 Distributed Algorithm for MD Assignment

In this algorithm the assignment of MD video and the construction of multiple multicast trees are performed in a distributed manner. Each node in the network will only select one video description to transmit it to its neighbor nodes. This condition is to ensure disjointness between multicast trees. Destination nodes are responsible to construct multiple node-disjoint paths to the multicast source, node S. Each destination node will select a number of disjoint paths equal to its preference number of MD video. If there are two paths have the same video description, the one with shortest delay will be chosen.

The source node S will broadcasts the information of the available MD video and the bandwidth requirements for each description to its neighbor nodes. Neighbor nodes that have enough bandwidth will randomly choose one description and rebroadcasts it along with its bandwidth requirement to its neighbor nodes. As we mentioned previously, each node will only choose one description to transmit it to its neighbor nodes to maintain disjointness between multicast trees. This process will continue to reach a destination node. When a destination node receives information about a video description, it will rebroadcast this information to its neighbor nodes. This means also that a destination node could be a forwarder node. If this destination node has enough bandwidth it will select another description to receive. After a destination node selects its proper paths it will send this information to the source node.

After the multicast source S receives the paths for each destination node, it constructs multiple node-disjoint multicast trees. To do so, nodes that have the same video description should be added to the same tree. Algorithm 5 describes the construction of multiple multicast trees.

```
1. for $i$ = 1 to $V$ do
2.   if node(i) has the 1st video description then
3.     Add node(i) to tree $t_1$
4.   else if
5.     Add node(i) to tree $t_2$
6.   end if
7. end for
```

Algorithm 5 Multiple multicast trees construction.

Figure 3.2 shows an example of MD video assignment and construction of multiple multicast trees. The multicast source S broadcasts information about two video descriptions (VD_1, and VD_2) to its neighbor nodes, nodes W, and Z. Each node will randomly select one video description to rebroadcast. Therefore, node W selects VD_1 and node Z selects VD_1. After that, nodes W and Z will rebroadcast this information to their neighbors nodes, nodes B, and C. This process will continue until this information reached the destination nodes, nodes R_1, R_2, and R_3. Destination nodes R_1, and R_3 will select the paths $S \rightarrow W \rightarrow B \rightarrow R_1$, and $S \rightarrow Z \rightarrow C \rightarrow R_3$, respectively, to receive VD_1. The destination node R_2 receives two paths with the same description, description VD_1. Therefore, it will select the path with minimum delay. Assume the path $S \rightarrow W \rightarrow B \rightarrow R_2$ is selected. Note that destination node R_2 receives the same video description through different paths. This can be related to the randomness of choosing a video description. Finally, the multicast source S will construct only one multicast tree using algorithm 5. Figure 3.2(c) shows multicast tree t_1.

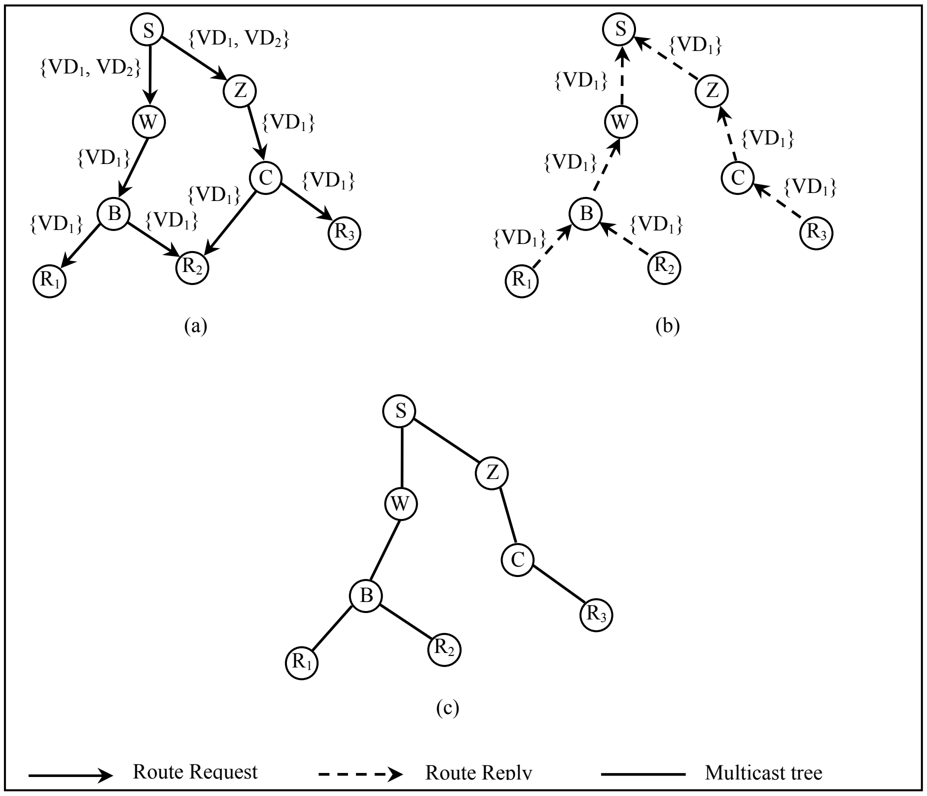


Figure 3.2 Distributed algorithm: (a) Route Request broadcasts. (b) Route Reply unicast. (c) Multicast tree construction.

3.5 Centralized Algorithm for MD Assignment

Before the construction of multiple node-disjoint multicast trees and the assignment of MD video, the multicast source *S* starts with constructing individual Multiple Node-Disjoint Paths (MNDP), with minimum delay, to each destination in the multicast group to meet the number of video descriptions required.

Definition 2: MNDP problem: consider a network represented by a graph $G = (V, E)$ and a bandwidth constraint w, find MNDP, set P, from the multicast source node S to the destination node R_i such that:

1. $d(p_{R_i})$ is minimize, $\forall p_{R_i} \in P$.
2. $Bw(p_{R_i}) \geq w, \forall p_{R_i} \in P$.

Algorithm 6 describes how MNDP are constructed. Before constructing multiple node-disjoint paths to each destination, we first remove all links with capacity less than the bandwidth requirement, and then we construct multiple shortest paths (in terms of delay) on the residual network. Based on the sets of MNDP constructed, then multicast heuristic algorithm constructs Multiple Node-Disjoint Multicast Trees (MNDMT) for the video transmission, as shown in Algorithm 7.

```
$P = \phi$  /* MNDP set */
**For** each destination $R_i$ **do**
    Let $G^*$ be equal to $G$
    **Repeat**
            Find a shortest path $p$ to $R_i$ (in terms of delay) in $G^*$ such that $Bw(p_i) \geq w$
            Add $p$ to $P$
            Remove all forwarding nodes of $p$ in $G^*$
    **Until**
            The number of paths in $P$ equal to the number of video descriptions required
```

Algorithm 6 MNDP.

```
**Step 1**) for $i = 1$ to $m$, find the set of MNDP $P_i$ by algorithm 6.
**Step 2**) Find a set $P_i$ that has the maximum number of paths.
**Step 3**) initially, let $T = P_i$, i.e., $t_1 = p_{i1}$, $t_2 = p_{i2}$, ..., $t_L = p_{iL}$.
**Step 4**) For $i = 2$ to $m$, add each path in $P_i$ to $T$ as follows:
        (i)     Find a path $p_{ij} \in P_i$ such that it intersects a tree, $t_k \subset T$ not covering
                $R_i$
                with the most links, and add $p_{ij}$ to $t_k$.
                        $t_k \leftarrow t_k + p_{ij}$
        (ii)    Remove $p_{ij}$ from $P_i$.
                        $P_i \leftarrow P_i - p_{ij}$
        (iii)   Repeat Steps (i) and (ii) until $P_i = \phi$.
```

Algorithm 7 MNDMT.

As a simple example, we consider the partial network topology in Figure 3.3(a), with a requirement of two descriptions for destination R_2 and one description for both destinations R_1 and R_3, to demonstrate the construction of multiple multicast trees. According to Algorithm 6, there are three path sets (Figure 3.3(b)) P_1, P_2, and P_3 from the source S to the destinations R_1, R_2, and R_3, where

$P_1 = \{p_{11}\} = S \to W \to B \to R_1$,

$P_2 = \{p_{21}, p_{22}\} = \{S \to W \to B \to R_2, S \to Z \to C \to R_2\}$

$P_3 = \{p_{31}\} = \{S \to Z \to C \to R_3\}$.

In Figure3.3(c)-(e), we show an example of multiple multicast trees construction using MNDMT. According to Algorithm 7, Step 2), the destination R_2 has the maximum number of paths, which is set P_2, (two paths); therefore we have two multicast trees according to step 3), namely, $t_1 = p_{21}$ and $t_2 = p_{22}$ as seen in Figure3.3(c). The path p_{11} of the destination R_1 will be added to t_1 (Figure 3.3(d)), according to Step 4(i), since it intersects t_1 with the most links. Because $P_1 = \phi$, then the algorithm picks up the next destination, R_3, and adds its path p_{31} to tree t_2 (Figure3.3(e)) according to Step 4(i). Since all the paths of each destination have been added, then the algorithm ends.

After constructing multiple multicast trees, Algorithm 8 assigns different video description to each tree. Therefore, trees t_1 and t_2 are assigned the first and second descriptions, respectively. Since any description can reproduce the original video signal, this we referred to as *independent-description* property of MDC, therefore the destination R_3 will be able to reproduce the original video signal. It is worth noting that if LC technique is used instead of MDC, and according to Chen-LC algorithm, therefore all the destinations will be only on the first or on the second tree. Thus, they will be only assigned the basic layer.

```
**For** i = 1 to L /*$L$ is the number of the multicast trees constructed*/
    **For** j= 1 to n /*$n$ is the number of MD video, ($L \le n$).*/
        **If** $Bw(t_i) \ge Bw(VD_j)$ **then**
            $VD_j \to t_i$
```

Algorithm 8 MD video assignment.

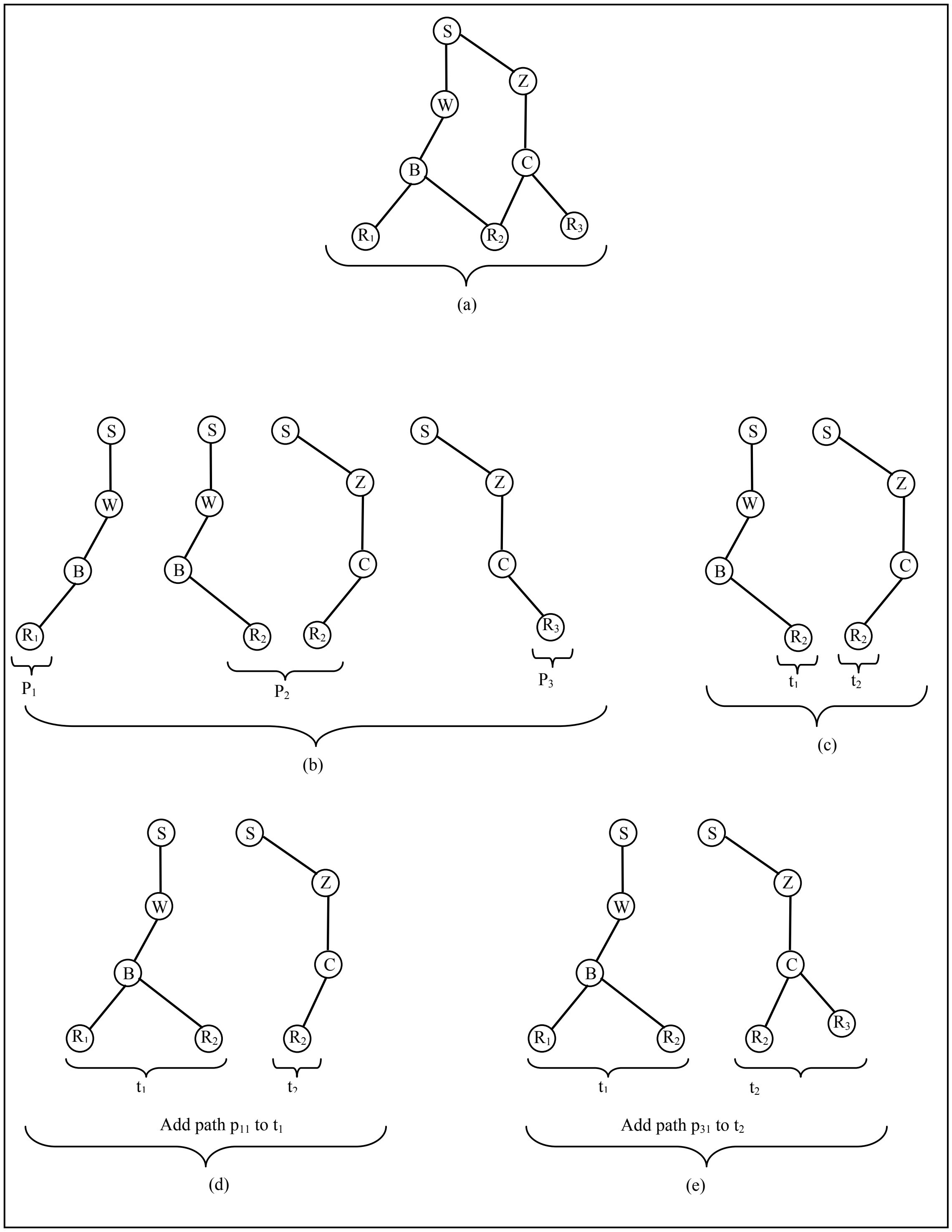


Figure 3.3 Demonstration of algorithm 6.

3.6 Performance Evaluation

This section deals with the performance evaluation of our developed algorithms. In particular, we evaluate the performance of our proposed algorithms, namely, Serial MDC, Distributed MDC, and Centralized MDC, and compare them with the algorithm proposed by Chen et al. in (Chen, Chan et Li, 2004). Chen et al. have proposed this algorithm for assigning a number of video layers that are encoded using LC technique; we referred to as Chen-LC. To make a fair comparison, we modified Chen-LC algorithm to construct node-disjoint multicast trees. Moreover, in order to take the bandwidth requirements for MDC and LC into consideration, we consider the video sequences reported in (Gogate et al., 2002). Since all the video sequences have roughly the same bit rate, we consider the video sequence of “Football”. The average video source rate is 1.5 Mbps for each description, whereas the average video source rate for the layered coder is 1.57 Mbps for the base layer and 1.45 Mbps for the enhancement layer.

We generate a wireless ad hoc network by placing a number of nodes at random locations in a square area of $1000\times1000m^2$. The radio transmission range is 250 m and the number of video descriptions required by each destination is uniformly distributed to be$\in\{1,2\}$. The residual bandwidth of each link is randomly chosen from [2, 10] Mbps. The delay in each link is randomly chosen from [1, 20] ms. Moreover, the multicast source S and a set of destinations $\mathbf{Y}$ are randomly chosen from the network graph to form a multicast session. Any destination node is at least 2-hop away from the multicast source.

We evaluate the performance of our proposed algorithm based on a partial topology. The partial topology is identified by logical levels with a multicast source node on the level zero, its 2-hop neighboring nodes on the first level, its 3-hop neighboring nodes on the second level and so on. In addition, links between nodes on the same level will be filtered out. Thus, the multicast source knows multiple paths to each destination node.

For each simulation, several experiments have been run to ensure 95% confidence interval. The 95% confidence intervals are always plotted, when they are not visible it means that they are smaller than the curve markers. To show the significance of our developed algorithm, we evaluate and compare its performance with the well-known multicast algorithm, Chen-LC, using the following metrics:

- **User's satisfaction Ratio (USR)**: This metric is defined as the total number of the assigned video descriptions to all destinations divided by the total number of requested video descriptions by all destinations. This metric presents the effectiveness of a protocol.

$$USR = \frac{\sum_{i=1}^{m} \mathbf{N}_{asg}(R_i)}{\sum_{i=1}^{m} \mathbf{N}_{req}(R_i)} \qquad (8)$$

where $\mathbf{N}_{asg}(R_i)$, and $\mathbf{N}_{req}(R_i)$, are the number of the assigned and requested video descriptions of the destination R_i respectively, and *m* is the number of destinations.

- **Number of pure forwarders (PF)**: It is defined as the number of pure forwarders on the multiple multicast trees that are not destinations. This measures the efficiency in terms of minimizing the number of pure forwarders nodes.

$$PF = \sum_{i=1}^{N} X_i \qquad (9)$$

where N is the network size and X_i is defined as:

$$X_i = \begin{cases} 1 & if\ X_i \in \mathbf{T} - \{S, \mathbf{Y}\} \\ 0 & otherwise \end{cases}$$

where $\mathbf{T} = t_1 \cup t_2$, S is the multicast source, and $\mathbf{Y}$ is the destinations set.

- **Bandwidth utilization (BwU)**: This metrics defined as the total used bandwidth for the video distribution trees.

$$BwU = \sum_{e \in u} Bw(e) \qquad (10)$$

where e denotes a link, u is the set of used links, and Bw denotes the bandwidth devoted to video distribution in link e.

- **Aggregate tree delay**: It represents the longest delay from the multicast source S to a destination R_i on the aggregate tree $\mathbf{T}$, as seen in Equation 4.

3.6.1 Varying Number of Multicast Destinations

Figure 3.4 to Figure 3.7 illustrate Serial MDC, Distributed MDC, Centralized MDC, and Chen-LC performance with varying number of multicast destination nodes while the network size is set to 50 nodes.

Figure 3.4 shows that Serial MDC, Distributed MDC, and Centralized MDC achieve higher user's satisfaction compared to Chen-LC. This can be related to the *independent-description* property of MDC. Serial MDC, Distributed MDC, and Centralized MDC are well scalable in term of number of destinations. Centralized MDC achieves a higher user satisfaction compared to Serial MDC and Distributed MDC. As the number of destinations increases, user's satisfaction decreases gradually. However, the user's satisfaction of Chen-LC decreases sharply as the number of destinations increases. That is, as a result of the *dependent-layer* property of LC.

Figure 3.5 depicts the number of pure forwarders nodes as a function of number of destination nodes. It can be seen that Centralized MDC has slightly higher number of pure forwarders nodes compared to the other algorithms. However, Distributed MDC has a lowest number of pure forwarders nodes.

In Figure 3.6, we plot the average bandwidth utilization. Clearly, the bandwidth utilization of Centralized MDC is slightly higher than the bandwidth utilization of the other algorithms. This is because Centralized MDC requires more number of pure forwarders, compared to the other algorithms; to constructs multiple node-disjoint trees (see Figure 3.5). However, Distributed MDC requires a minimum bandwidth for video distribution trees. This is because Distributed MDC has a minimum number of pure forwarders nodes.

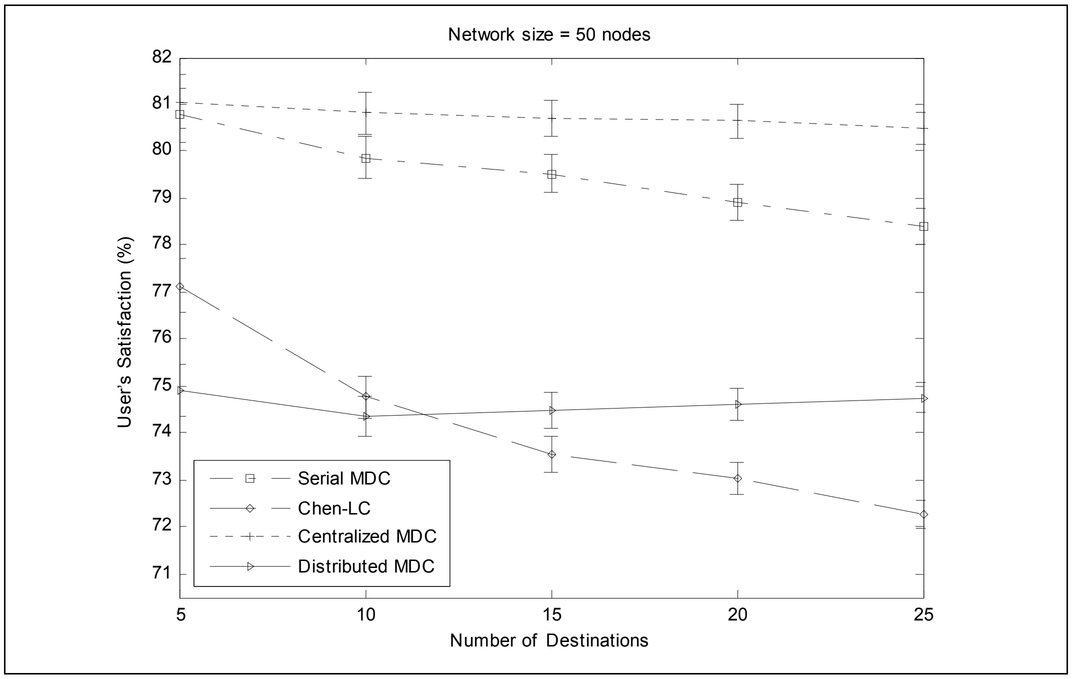


Figure 3.4 User satisfaction vs. Number of destinations. Network size = 50 nodes.

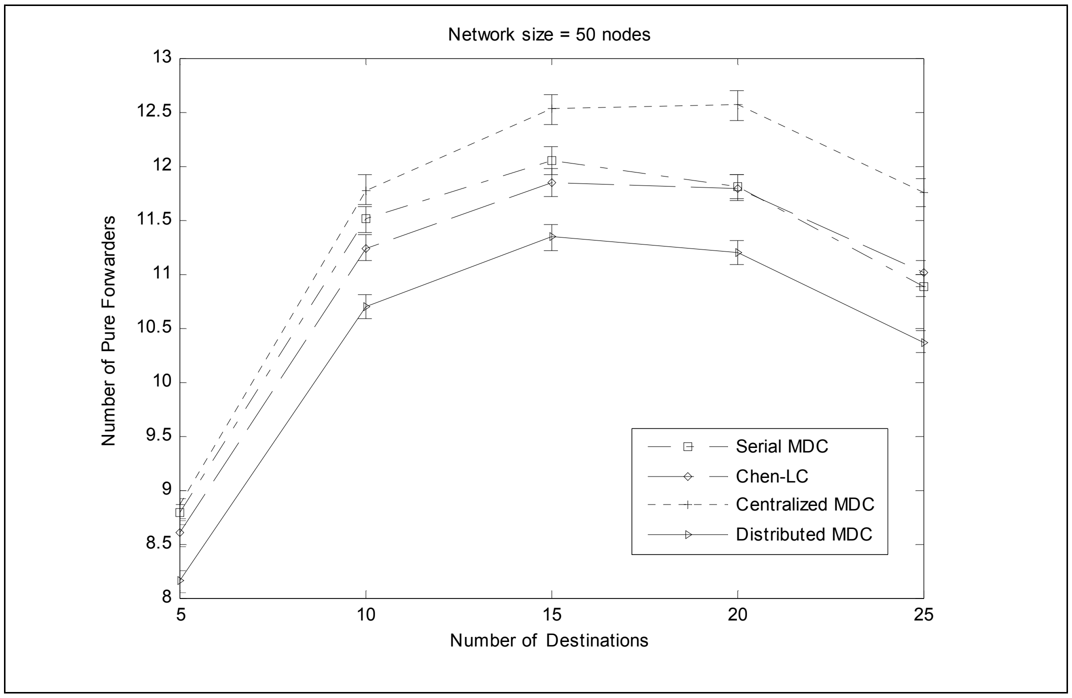


Figure 3.5 Number of Pure Forwarders vs. Number of destinations. Network size = 50 nodes.

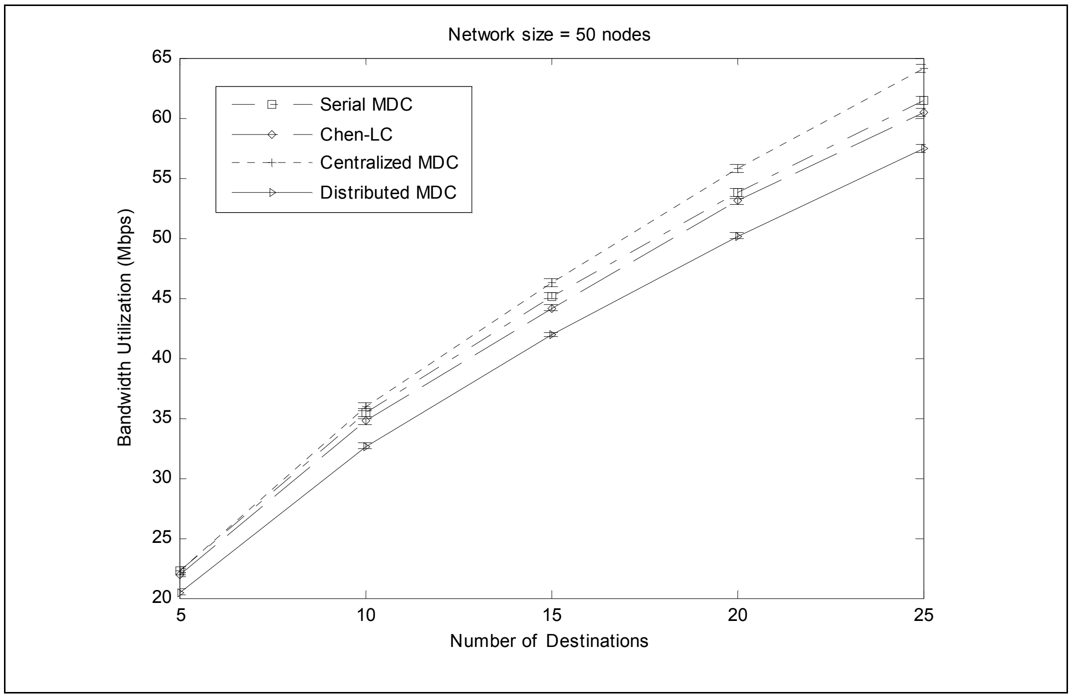


**Figure 3.6 Bandwidth utilization vs. Number of destinations.
Network size = 50 nodes.**

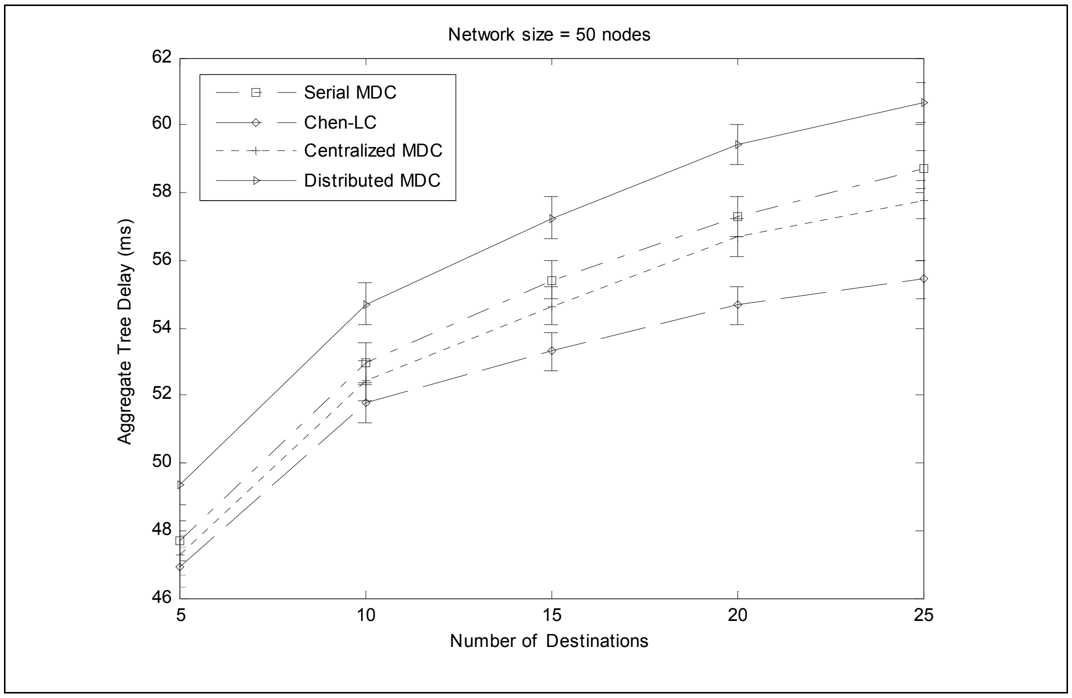


**Figure 3.7 Aggregate tree delay vs. Number of destinations.
Network size = 50 nodes.**

We show in Figure 3.7 the aggregate tree delay as a function of number of destinations. All algorithms achieve a comparable delay as compared to each other. As the number of destinations increases the aggregate tree delay increases. This is because more paths are constructed to build multiple multicast trees.

Figure 3.8 to Figure 3.11 illustrate Serial MDC, Distributed MDC, Centralized MDC, and Chen-LC performance with varying number of multicast destination nodes while the network size is set to 100 nodes. As the network size increase from 50 nodes (Figure 3.4) to 100 nodes (Figure 3.8), the user's satisfaction for all algorithms increases. This is because the number of nodes in the network increases. As a result, the number of paths to each destination is increased. Again, Serial MDC, Distributed MDC, and Centralized MDC show good scalability as the number of destinations increase.

As the number of nodes in the network increases from 50 to 100 nodes, the number of pure forwarders nodes, the bandwidth utilization, and the aggregate tree delay (in Figure 3.9, Figure 3.10, and Figure 3.11) are increased compared to Figure 3.5, Figure 3.6, and Figure 3.7, respectively.

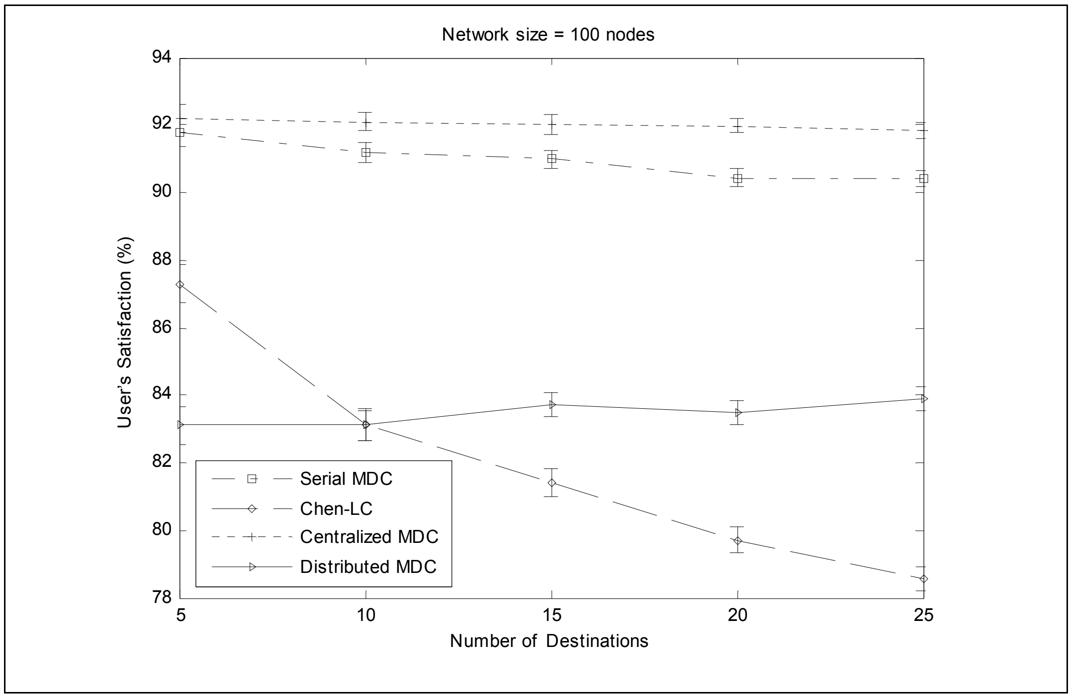


Figure 3.8 User Satisfaction vs. Number of destinations. Network size = 100 nodes.

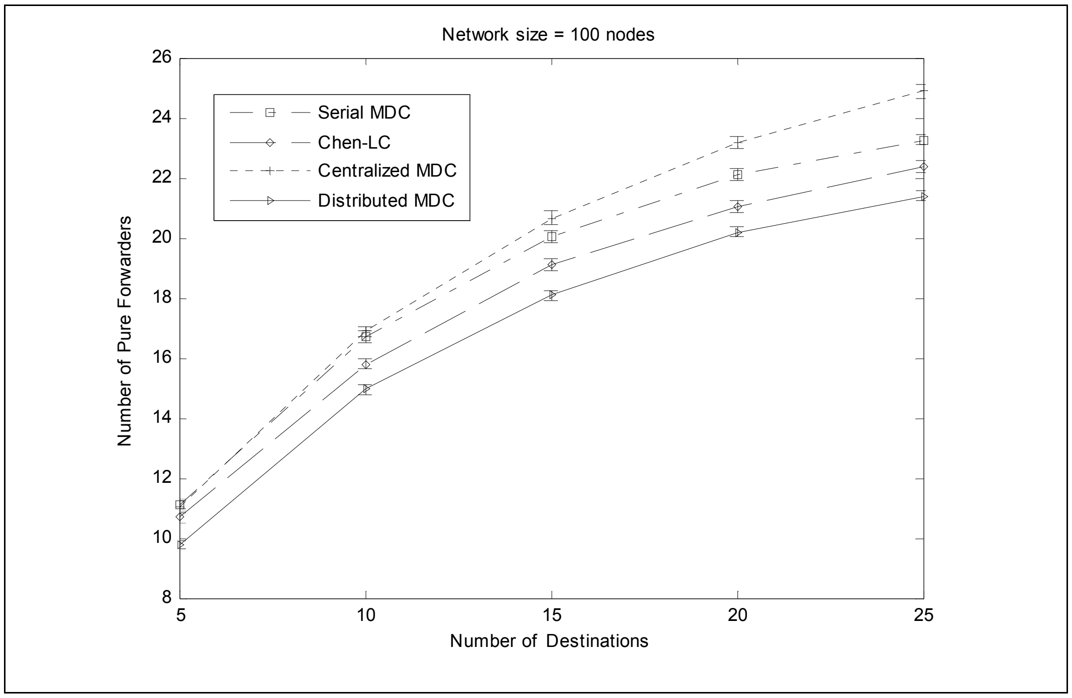


Figure 3.9 Number of Pure forwarders vs. Number of destinations. Network size = 100 nodes.

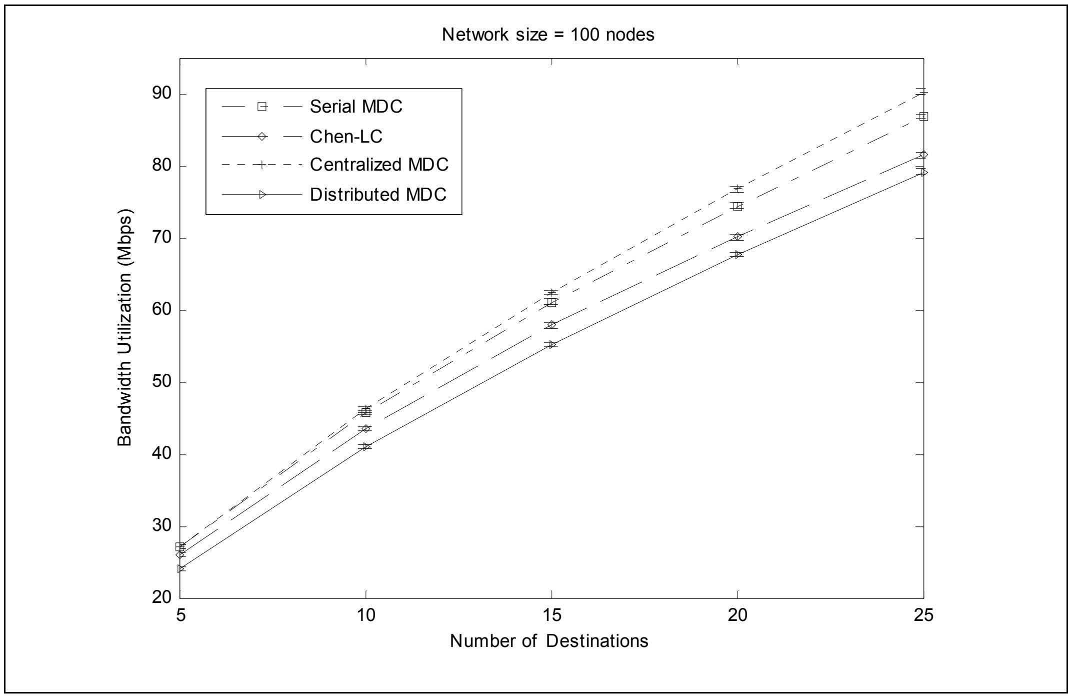


Figure 3.10 Bandwidth utilization vs. Number of destinations. Network size = 100 nodes.

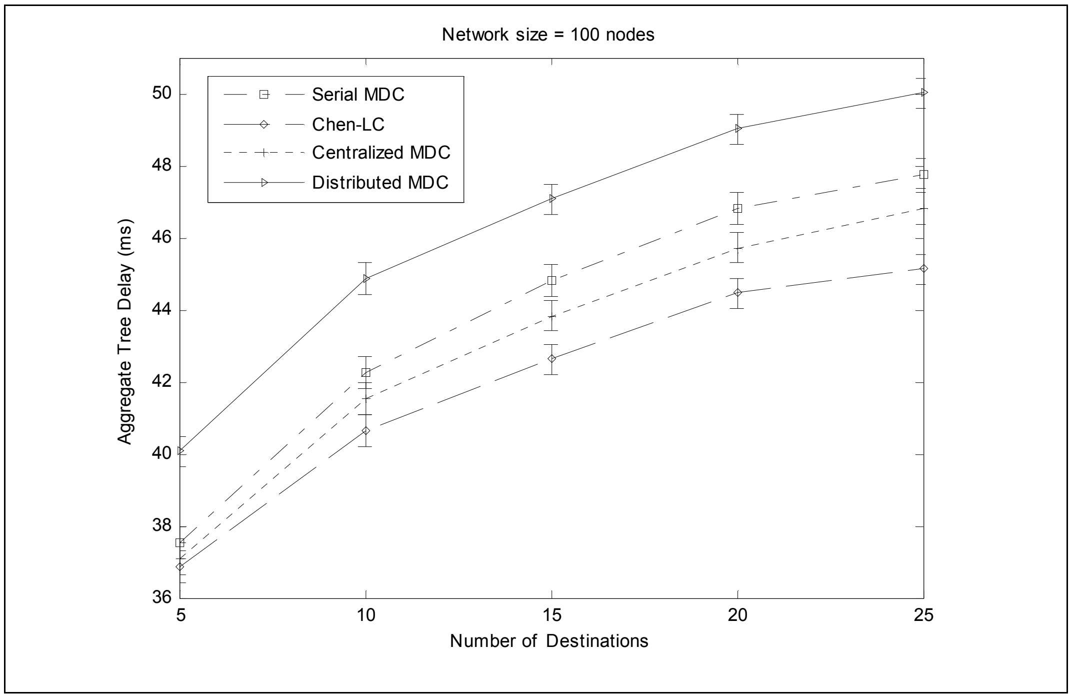


Figure 3.11 Aggregate tree delay vs. Number of destinations. Network size = 100 nodes.

3.6.2 Varying Network Size

Figure 3.12, Figure 3.13, Figure 3.14, and Figure 3.15 compare Serial MDC, Distributed MDC, Centralized MDC, and Chen-LC performance, with varying number of nodes in the network from 50 to 100 nodes, in terms of user satisfaction, number of pure forwarders, bandwidth utilization, and aggregate tree delay. The number of destinations is set to 10 and 30 nodes.

Centralized MDC achieve a higher user satisfaction (see Figure 3.12) compared to the other algorithms. The cost of that is the increase in the number of pure forwarders nodes (see Figure 3.13), the bandwidth utilization (see Figure 3.14), and the aggregate tree delay (see Figure 3.15). However this cost is still comparable. As the network size increases, the user satisfaction for all algorithms increases. We related that to the increase in the number of resources in the network, i.e., number of nodes, and bandwidth. Figure 3.15 shows that as the network size increases the aggregate tree delay decreases. This is because more alternate paths (with minimum delay) may exist.

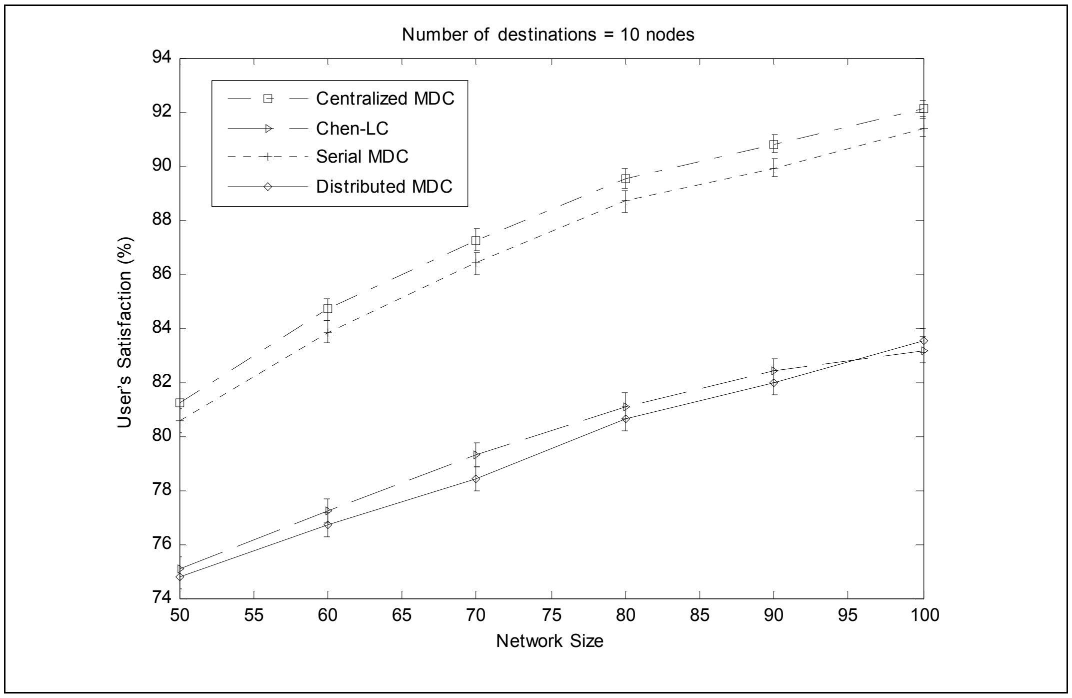


Figure 3.12 User satisfaction vs. Network size.
Number of destinations = 10 nodes.

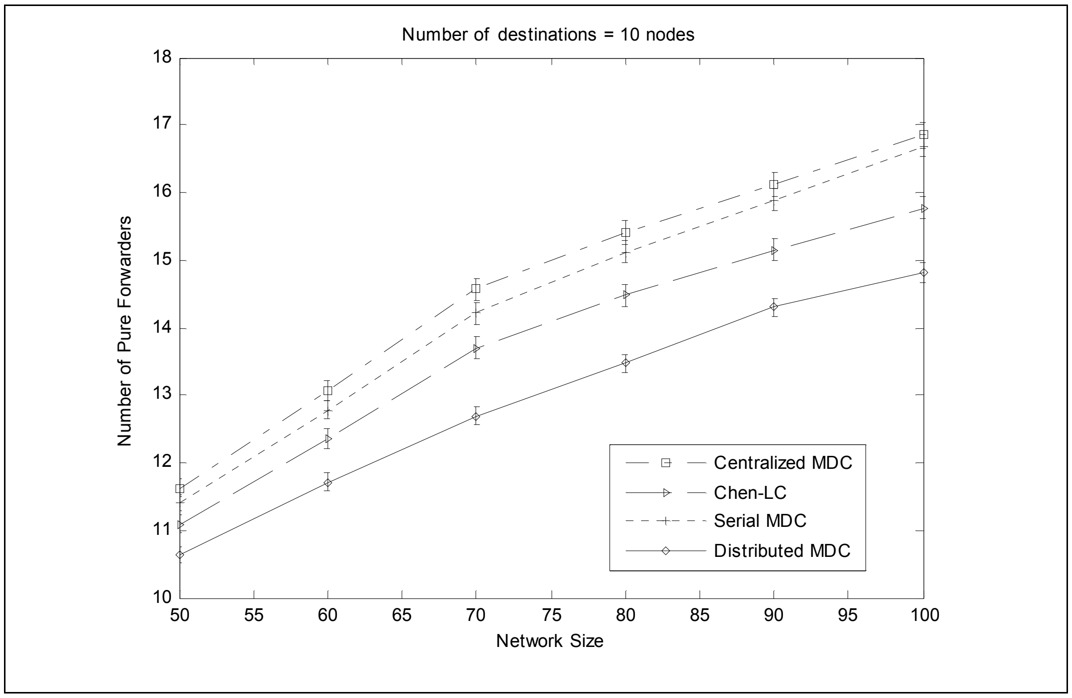


Figure 3.13 Number of pure forwarders vs. Network size. Number of destinations = 10 nodes.

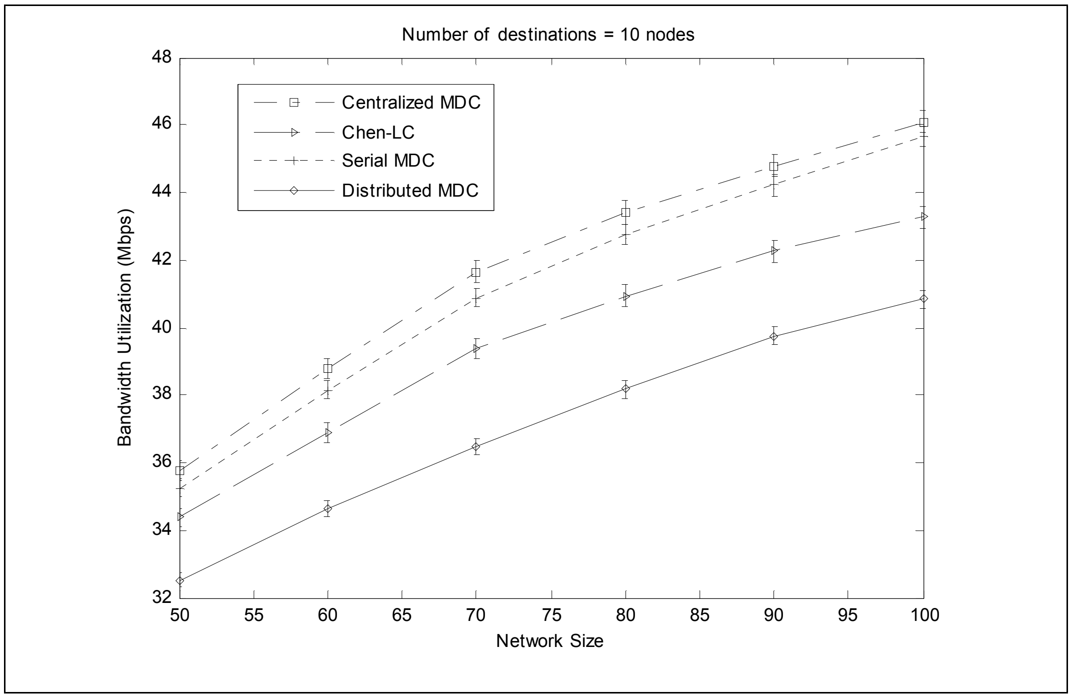


Figure 3.14 Bandwidth utilization vs. Network size. Number of destinations = 10 nodes.

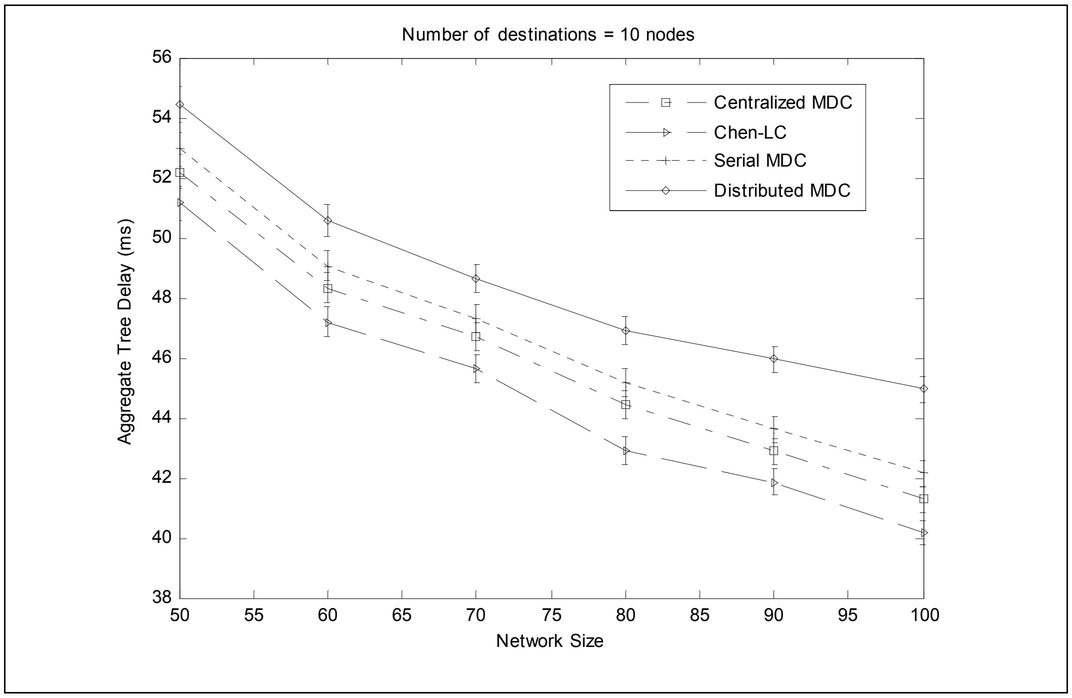


Figure 3.15 Aggregate tree delay vs. Network size. Number of destinations = 10 nodes.

In Figure 3.16 to Figure 3.19, the number of destinations increases from 10 to 30 nodes. We can note that the user satisfaction for all algorithms decreases compared to Figure 3.12. Centralized MDC algorithm still has a higher user satisfaction compared to the others.

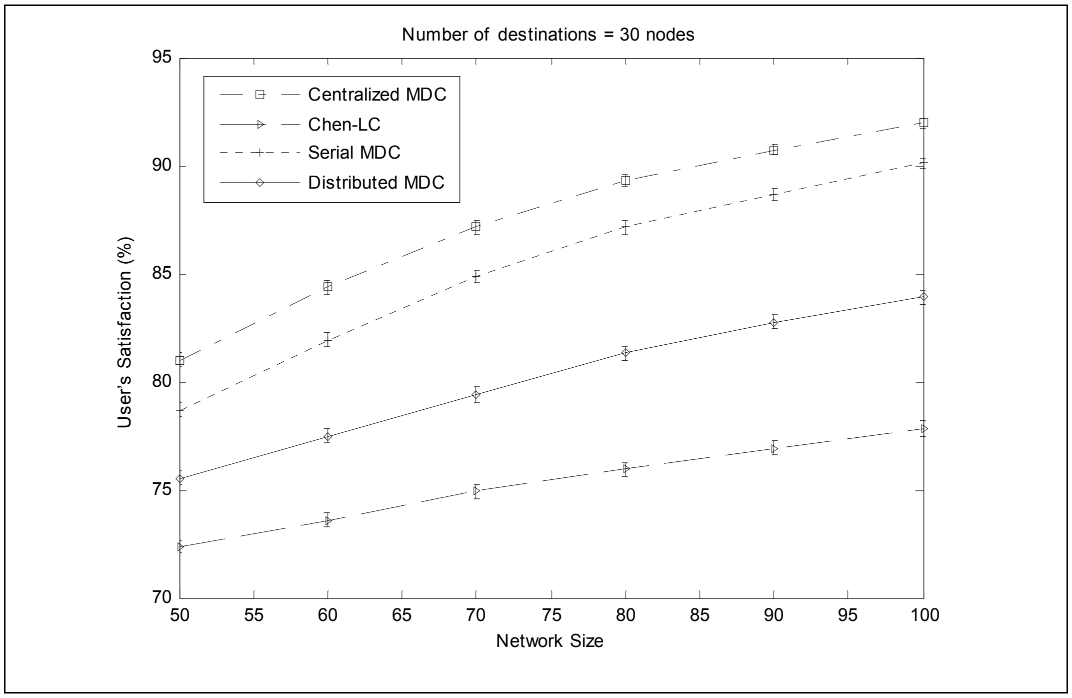


Figure 3.16 User satisfaction vs. Network size.
Number of destinations = 30 nodes.

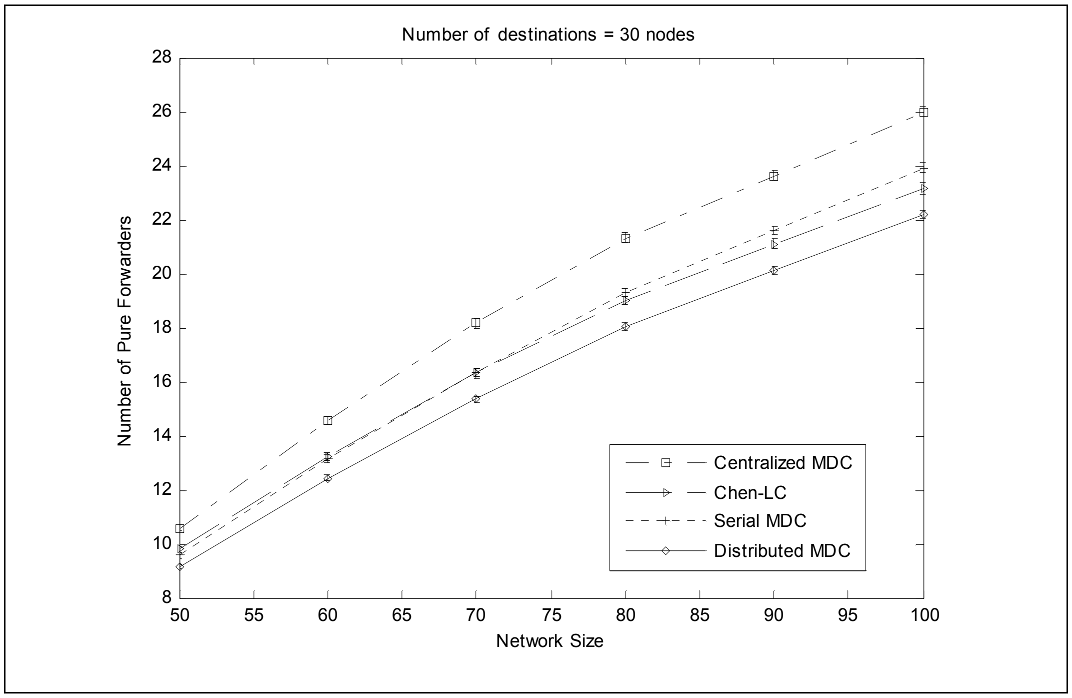


Figure 3.17 Number of pure forwarders nodes vs. Network size.
Number of destinations = 30 nodes.

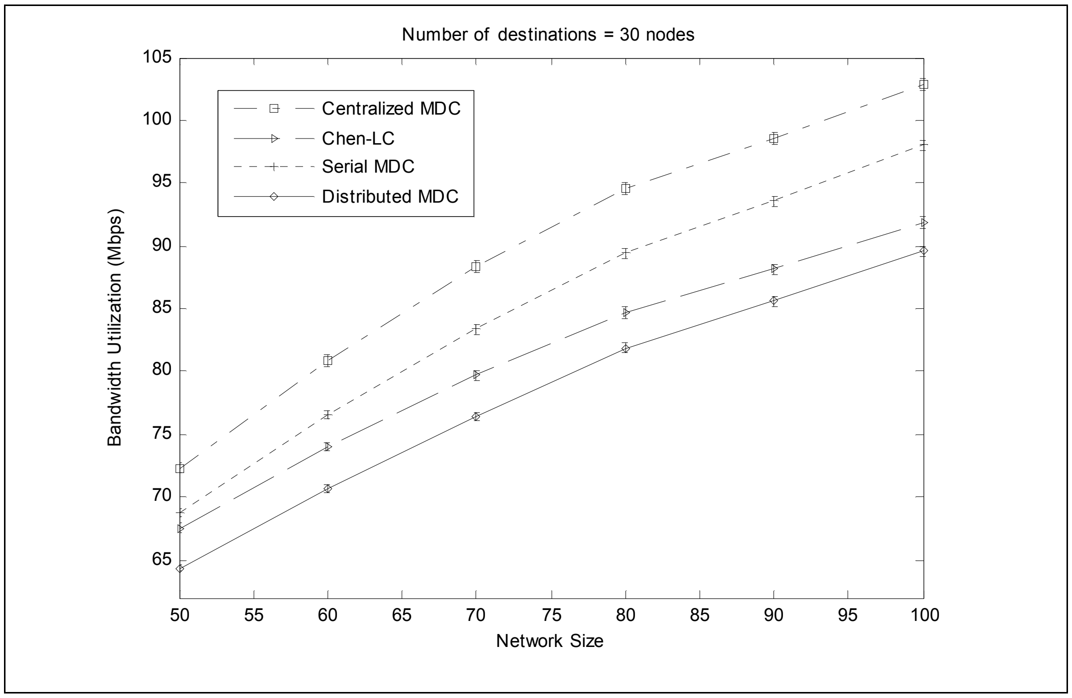


Figure 3.18 Bandwidth utilization vs. Network size. Number of destinations = 30 nodes.

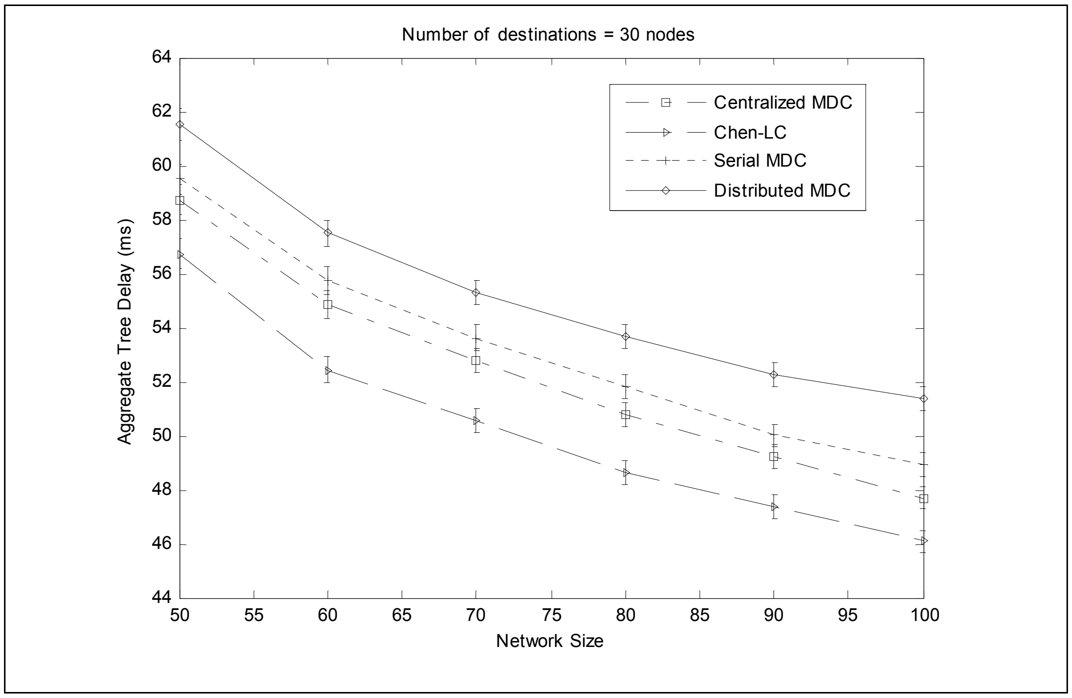


Figure 3.19 Aggregate tree delay vs. Network size. Number of destinations = 30 nodes.

3.7 Multiple Shortest Path Tree (MSPT) versus Multiple Steiner Minimum Tree (MSMT)

In this section we present two algorithms for constructing multiple multicast trees and assigning MD video. The two algorithms are: MSPT and MSMT algorithms. In contrast to Serial MDC, Distributed MDC, and Centralized MDC, MSPT and MSMT algorithms do not consider the *independent-description* property of MDC. Instead, they assign them in a sequential manner.

3.7.1 Types of Multiple Paths

There are two types of multiple paths, disjoint paths and non-disjoint paths. However, there are two types of disjoint paths: node-disjoint and link- disjoint. Node-disjoint paths, also known as totally disjoint paths, do not have any common nodes, except the source and destination. In contrast, link- disjoint paths do not have any common links, but may have common nodes. On the other hand, non-disjoint paths can have nodes and links in common. Refer to Figure 3.20 for examples of the different kinds of multiple paths.

Node-disjoint paths offer some advantages over non-disjoint and link-disjoint paths. In non-disjoint and link-disjoint paths, a single node failure can cause multiple paths that share that node to fail. In node-disjoint paths, a single node failure will only cause a single path to fail. Thus, node-disjoint paths offer the highest degree of fault-tolerance.

The main advantage of non-disjoint paths is that they can be more easily discovered. Because there are no limitations that require the paths to be link or node disjoint, more non-disjoint paths exist in a given network than link or node disjoint paths. Because node-disjointedness is a stricter requirement than link-disjointedness, node-disjoint routes are the least abundant and are the hardest to find. In moderately dense networks, there may only exist a small number of node disjoint routes between any two arbitrary nodes, especially as the distance between the nodes increases (Ye, Krishnamurthy et Tripathi, 2003). This is because there may be sparse areas between the two nodes that act as bottlenecks.

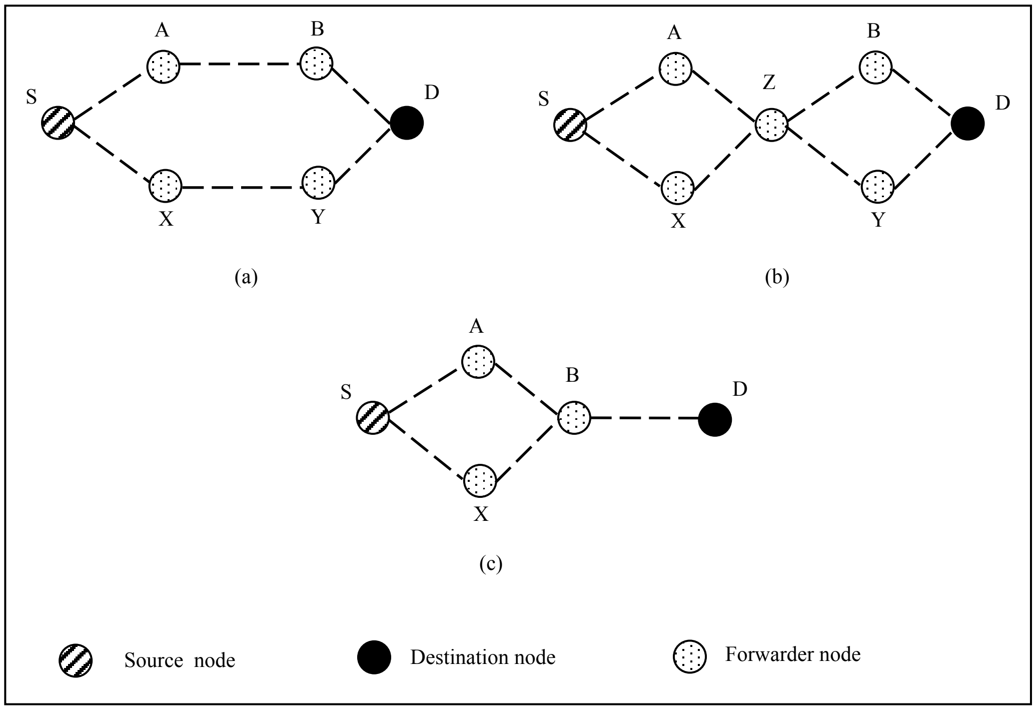


Figure 3.20 Different types of multiple paths:
(a) Node-disjoint. (b) Link-disjoint. (c) Non-disjoint.

3.7.2 Multiple Shortest Path Trees (MSPT) Algorithm

Shortest path tree (SPT) constructs a multicast tree with shortest path from a multicast source node to every destination node. SPT ensures that the end-to-end delay from the multicast source to each destination is the minimum. Single SPT that meet the destinations' requirement (the number of video layers required) may not exist, even though there are enough resources in the network. Thus, MSPT can greatly increases the number of video layers delivered to each destination.

3.7.3 Multiple Steiner Minimum Trees (MSMT) Algorithm

Steiner minimum tree (SMT) algorithm constructs a multicast tree that spans all the multicast group members with minimum number of links. SMT guarantees certain bound on the end-to-end delay of the multicast tree. The construction of the MSMT is based on the Steiner tree algorithm described in (Kou, Markowsky et Berman, 1981).

3.7.4 Simulation Setup

The simulation is performed using the following parameters: network size (the number of nodes in the network), the number of destinations in the network (multicast group size), and the transmission range is set to 250 *m*.

We generate network graphs in $1000 \times 1000\,m^2$ of a 2-D simulation area, by randomly distributing a certain number of nodes. Once the nodes are placed in the square area and their transmission ranges are decided (each node has the same transmission range), thus a network graph is generated where two nodes within each other's transmission range (the distance between them is less than the transmission range) will have a link. We assume that there are three types of nodes. The first type is capable of handling one video description (it has capacity of one), the second type is capable of handling two video descriptions (it has capacity of two), and the third one is capable to handle three video descriptions (it has capacity of three). The probability of generating a node with one, or two, or three is equal to 1/3. The multicast source and destinations are randomly chosen such that a multicast source has a capacity of three and the destination nodes are located at least two-hop away from the multicast source. The source node generates three video descriptions. The link bandwidth is randomly distributed to be $\in \{1, 2, 3\}$, where 1, 2, and 3 mean the link is capable of handling one, two, and three video descriptions, respectively.

In the next sections, we perform two groups of simulations. In the first group, we vary the number of destination nodes from 5 to 25 nodes and we fix the network size to 50 and 100 nodes. In the second group, we vary the network size from 50 to 100 and we set the multicast group size to 10 and 30 nodes. For both groups of simulation, the proposed algorithms are compared in terms of the following metrics:

- **User satisfaction:** refer to Equation 8.
- **Number of forwarders nodes:** number of forwarding nodes defined as the number of nodes on the multicast trees except the multicast source and the leaf destinations.

3.7.5 Multiple Node-Disjoint Multicast Trees

In multiple node-disjoint trees (totally disjoint trees), there are no common nodes between the multicast trees except the source and destinations. In other words, all paths to each destination are totally node-disjoint. The first multicast tree is composed of all nodes that have the first video description (MDC_1). The second multicast tree is composed of all nodes that have the second video description (MDC_2). And finally, the third multicast tree is composed of all nodes that have the third video description (MDC_3). Figure 3.21 shows the flow diagram of multiple node-disjoint multicast trees construction and MD video assignment.

In Figure 3.22 and Figure 3.23 we plot user satisfaction and number of forwarding nodes, respectively, for both algorithms MSPT and MSMT versus the number of destinations. The number of network size is set to 50 and 100 nodes. It is clear that user satisfaction drops steadily as the multicast group size increases. This is because the available network resources become less and user satisfaction therefore becomes lower no matter which algorithm is used. As the network size increase from 50 to 100 nodes; user satisfaction increases. This is because the available resources in the network are increased. Statistically speaking, both algorithms have the same user satisfaction.

Figure 3.23 shows that as the number of destinations increases; the number of forwarding nodes increases. In addition, as the network size increase from 50 to 100 nodes; the number of forwarding nodes increases. Both algorithms have the same number of forwarding nodes.

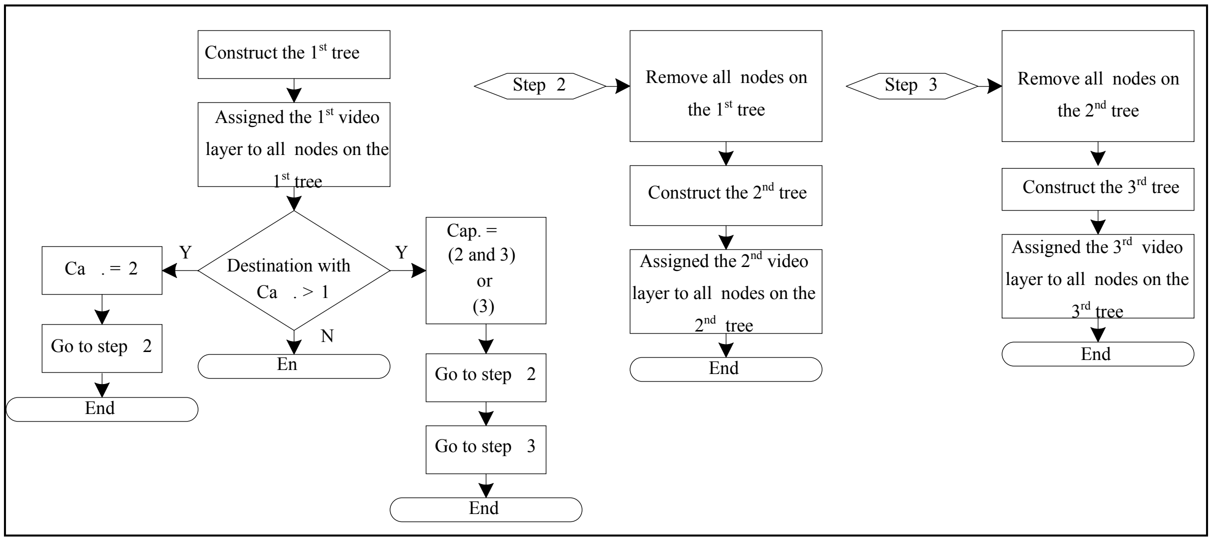

Construct the 1st tree
Assigned the 1st video layer to all nodes on the 1st tree
Destination with Ca . > 1
Y
Ca . = 2
Go to step 2
End
N
En
Y
Cap. = (2 and 3) or (3)
Go to step 2
Go to step 3
End
Step 2
Remove all nodes on the 1st tree
Construct the 2nd tree
Assigned the 2nd video layer to all nodes on the 2nd tree
End
Step 3
Remove all nodes on the 2nd tree
Construct the 3rd tree
Assigned the 3rd video layer to all nodes on the 3rd tree
End

Figure 3.21 Construction of multiple node-disjoint trees.

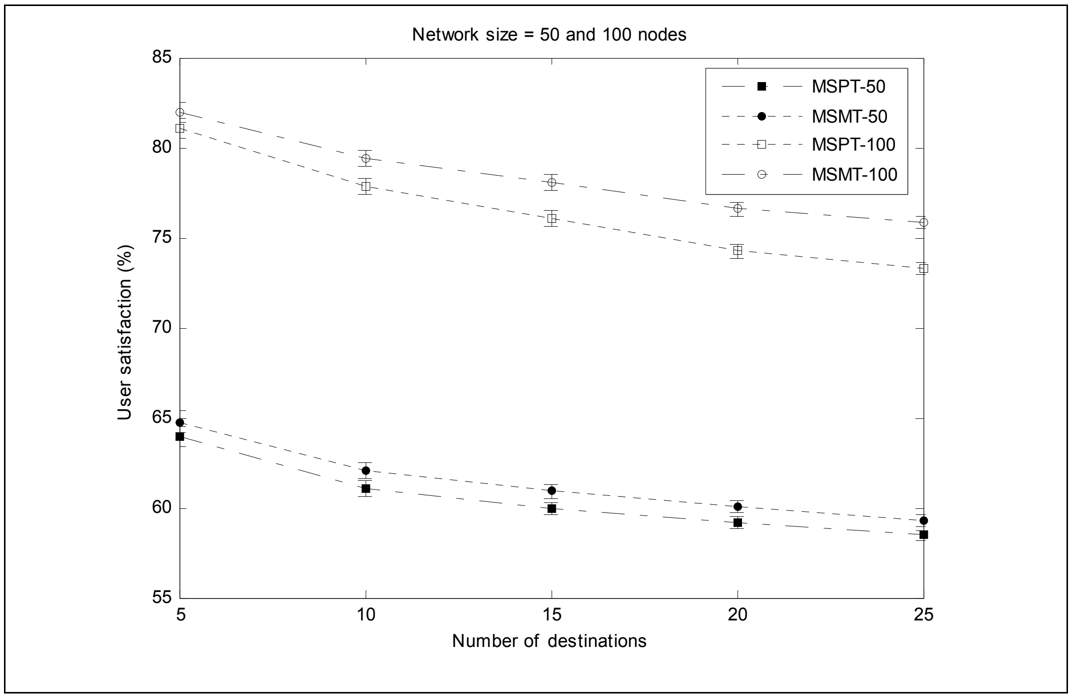


Figure 3.22 Multiple node-disjoint trees:
User satisfaction vs. Number of destinations.

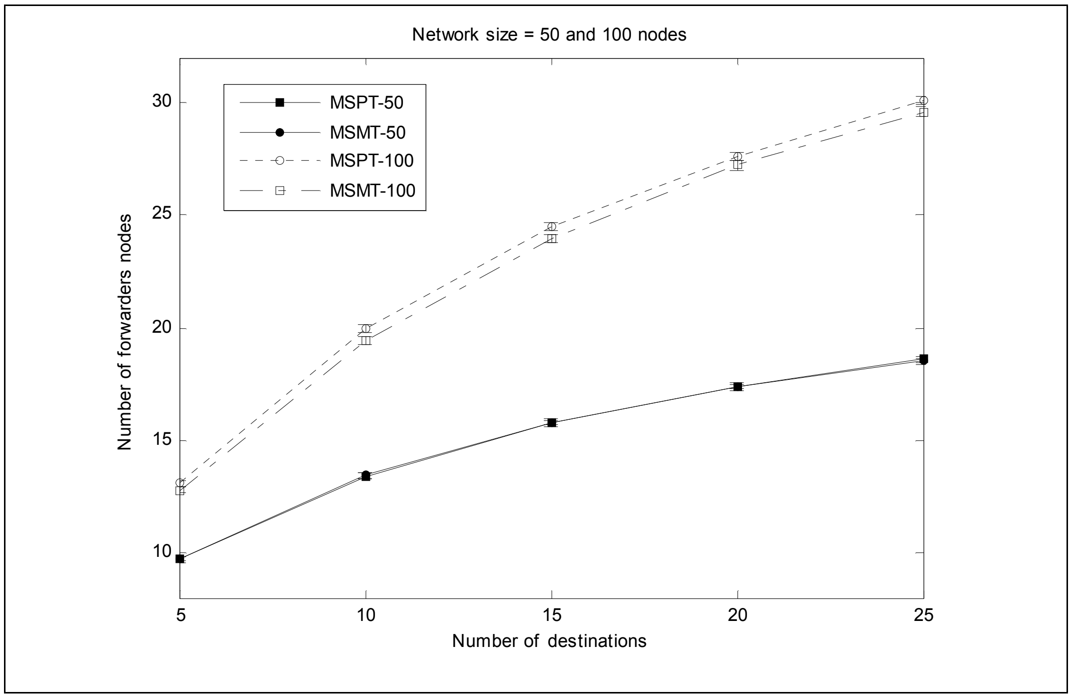


Figure 3.23 Multiple node-disjoint trees:
Number of forwarders nodes vs. Number of destinations.

Figure 3.24 plots user satisfaction versus the number of nodes in the network. Number of destinations is set to 10 and 30 nodes. For both algorithms, as the number of nodes in the network increases user satisfaction ratio increases. This is because the available network resources are increased. When the number of destinations increases from 10 to 30 nodes; user satisfaction decreases. Figure 3.25 shows that both algorithms have the same number of forwarders nodes. As the number of destinations increases from 10 to 30 nodes; the number of forwarders nodes increases.

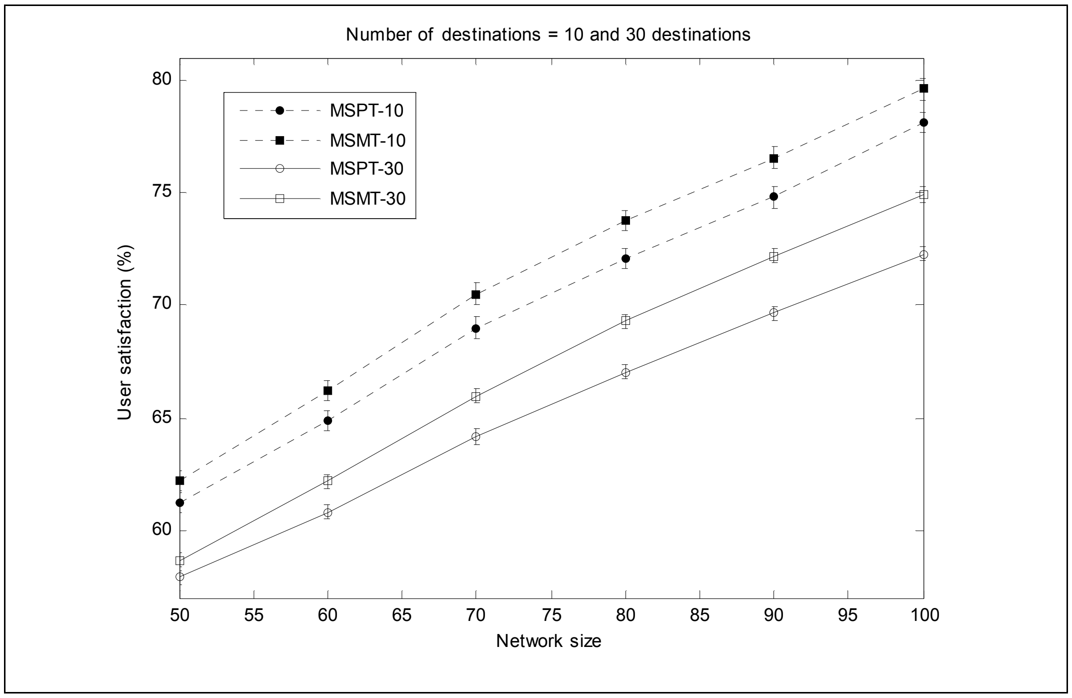


Figure 3.24 Multiple node-disjoint trees:
User satisfaction vs. Network size.

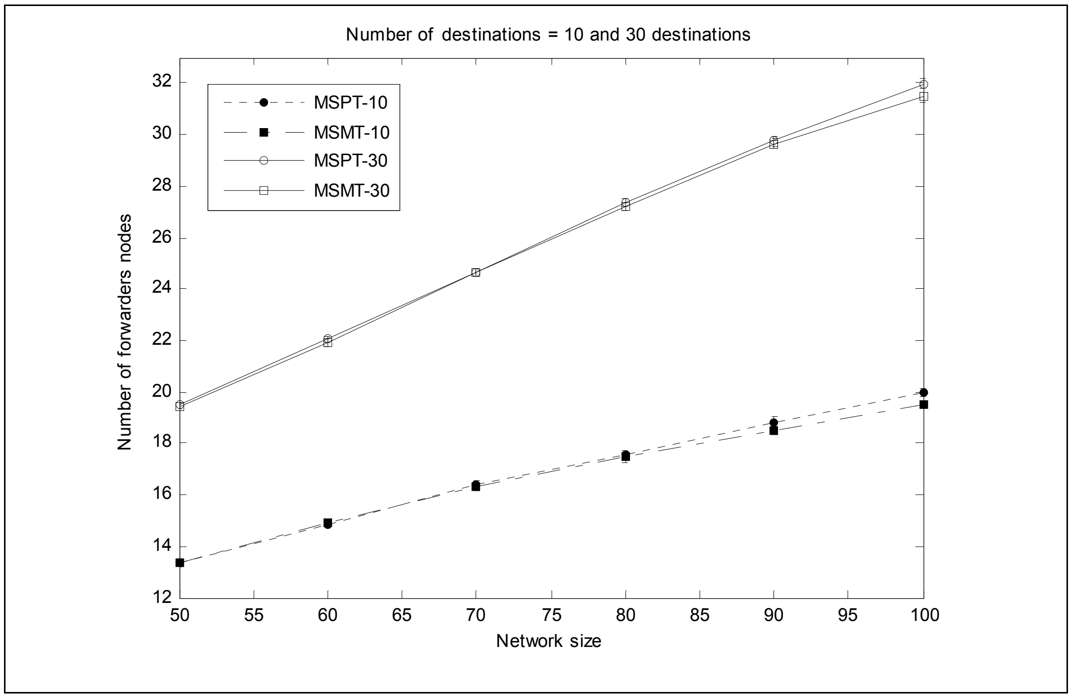


Figure 3.25 Multiple node-disjoint trees:
Number of forwarders nodes vs. Network size.

3.7.6 Multiple Hybrid-I Multicast Trees

In order to construct Hybrid-I multicast trees, we allow nodes to handle more than one description according to its capacities, but links are allowed to handle only one description. Therefore, link and node disjoint paths may be appeared. This will depend on the aggregate resources of paths for each destination. On the other hand, non-disjoint paths will not be existed. Because our multiple multicast trees will have nodes in common and there are no links in common, we refer to this type of trees as Hybrid-I. Figure 3.26 shows the flow diagram of the construction of multiple Hybrid-I multicast trees and the assignment of MD video.

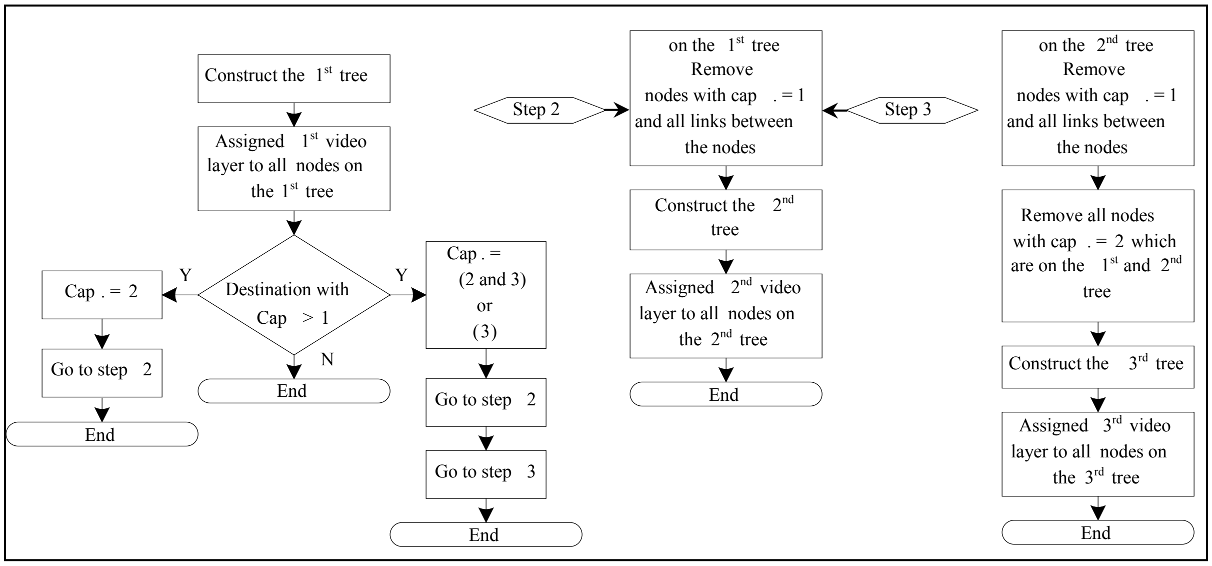

Construct the 1st tree
Assigned 1st video layer to all nodes on the 1st tree
Destination with Cap > 1
Y
N
Cap . = 2
Go to step 2
End
End
Cap . = (2 and 3) or (3)
Go to step 2
Go to step 3
End
Step 2
on the 1st tree Remove nodes with cap . = 1 and all links between the nodes
Construct the 2nd tree
Assigned 2nd video layer to all nodes on the 2nd tree
End
Step 3
on the 2nd tree Remove nodes with cap . = 1 and all links between the nodes
Remove all nodes with cap . = 2 which are on the 1st and 2nd tree
Construct the 3rd tree
Assigned 3rd video layer to all nodes on the 3rd tree
End

Figure 3.26 Construction of Hybrid-I multicast trees.

Figure 3.27 shows that user satisfaction increases as the number of network increases from 50 to 100 nodes. On the other hand, for a fixed number of nodes the network, it decreases as the number of destinations increases from 5 to 25 destinations. As the number of destinations increases, the number of forwarding nodes increases. When the number of nodes in the network decreases from 100 to 50 nodes; the number of forwarding nodes decreases. This is shown in Figure 3.28.

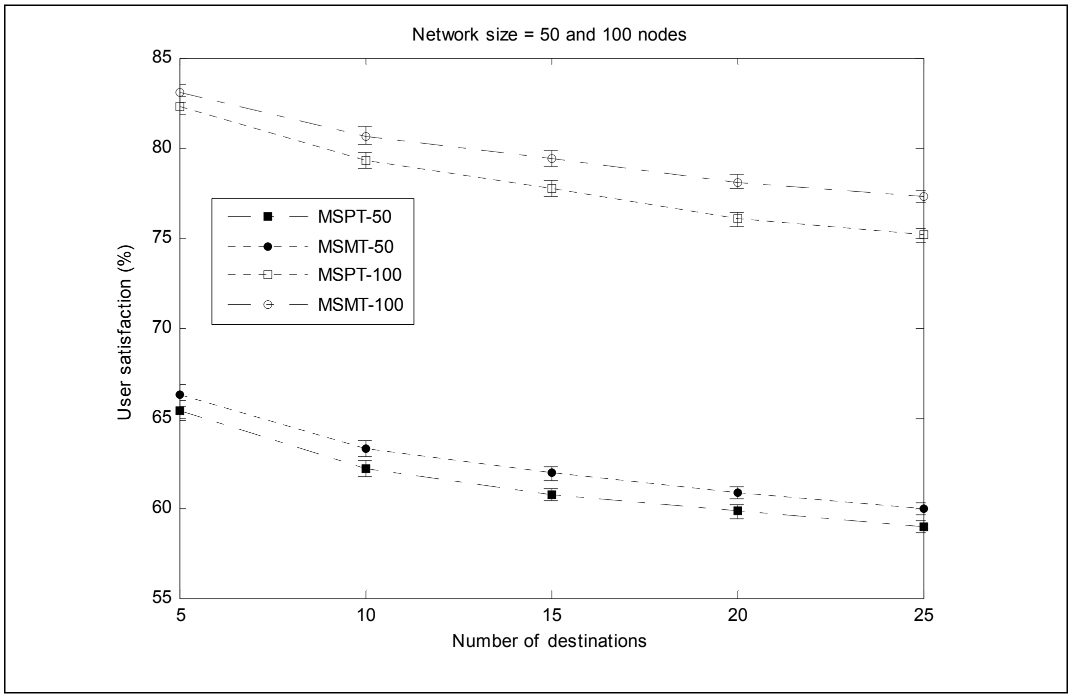


Figure 3.27 Hybrid-I multicast trees:
Users satisfaction vs. Number of destinations.

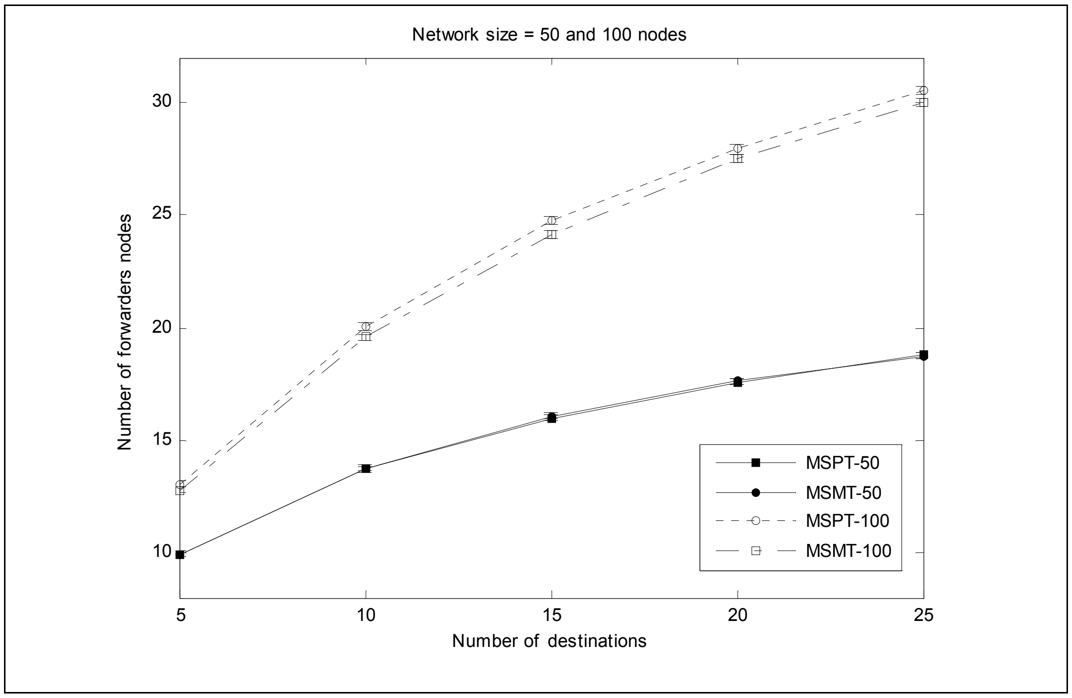


Figure 3.28 Hybrid-I multicast trees:
Number of forwarders vs. Number of destinations.

Figure 3.29 and Figure 3.30 show the change of user satisfaction and number of forwarders nodes, respectively, for two sets of destinations, 10 and 30, as the number of nodes in the network changes from 50 to 100 nodes.

It is clear, that user satisfaction increases as the number of nodes in the network increases. When the number of destination decreases from 30 to 10 destinations, user satisfaction increases, as depicted in Figure 3.29.

Figure 3.30 shows as the number of nodes in the network increases from 50 to 100 nodes the number of forwarding nodes increases for set of destinations, 10 and 30 destinations. This because the destinations becomes more sparse from the source, therefore more forwarding nodes are needed to connect them to the multicast tree. As the number of destinations decreases from 30 to 10 destinations, the number of forwarding nodes decreases.

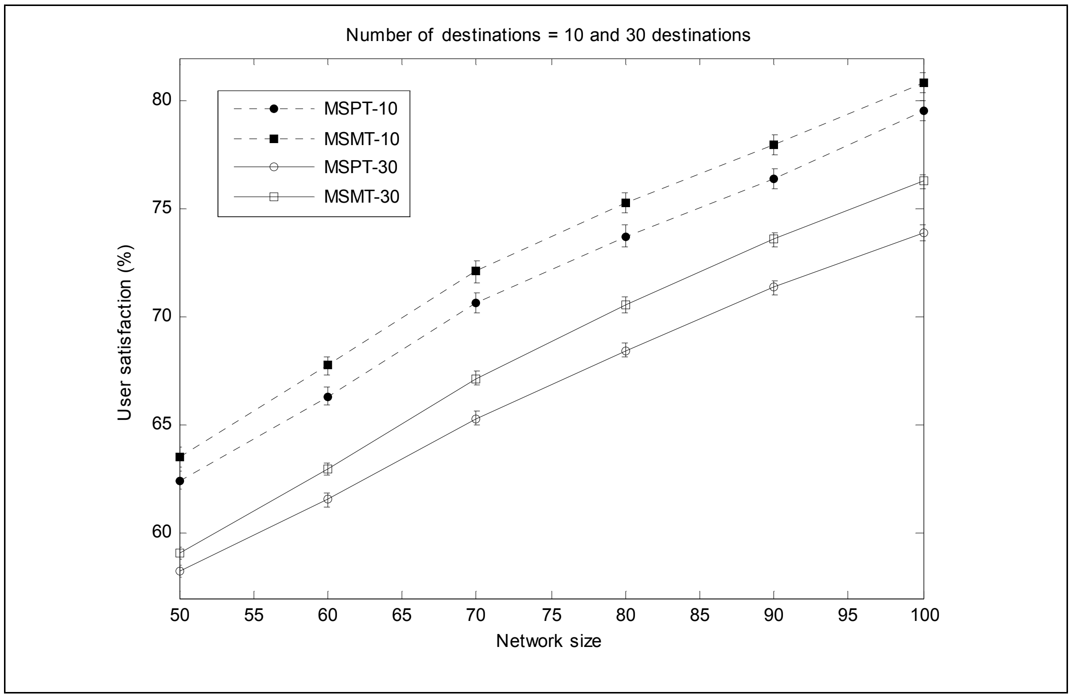


Figure 3.29 Hybrid-I multicast trees: User satisfaction vs. Network size.

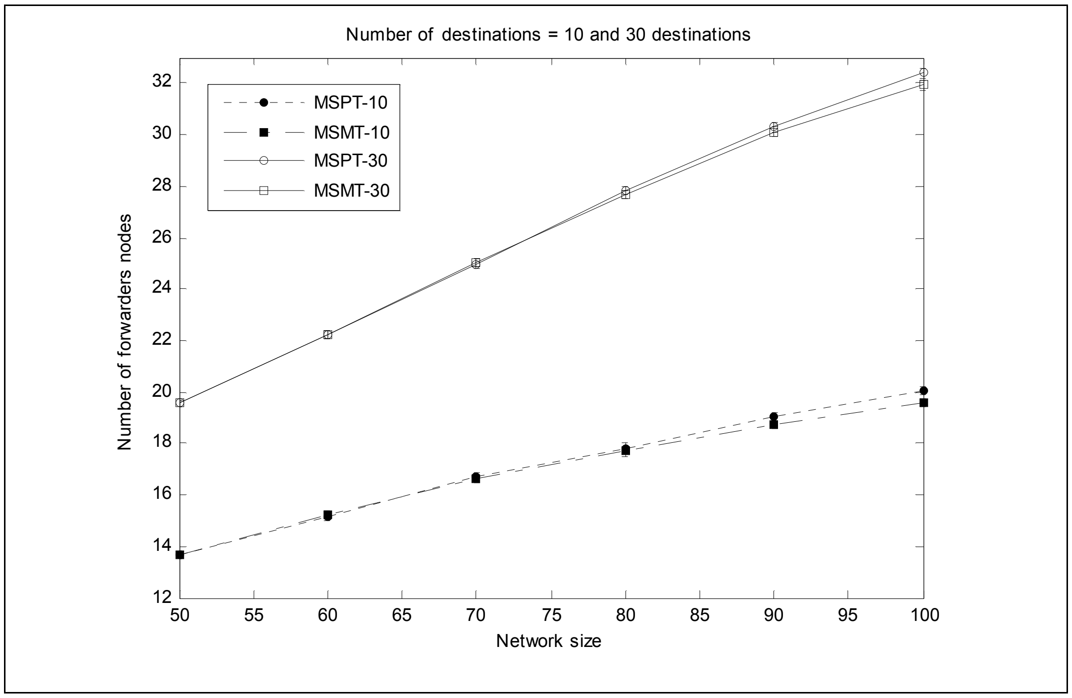


Figure 3.30 Hybrid-I multicast trees: Number of forwarders nodes vs. Network size.

3.7.7 Multiple Hybrid-II Multicast Trees

In order to construct Hybrid-II multicast trees, we allow nodes and links to handle more than one description according to their capacities and available bandwidth, respectively. Therefore, non-disjoint paths, node and link disjoint paths may be existed. Again, this will depend on the aggregate resources of paths for each destination. Thus, the resulting multiple multicast trees may have nodes and links in common. We called this type, Hybrid-II. Figure 3.31 shows the flow diagram of the construction of multiple Hybrid-II multicast trees and the assignment of MD video.

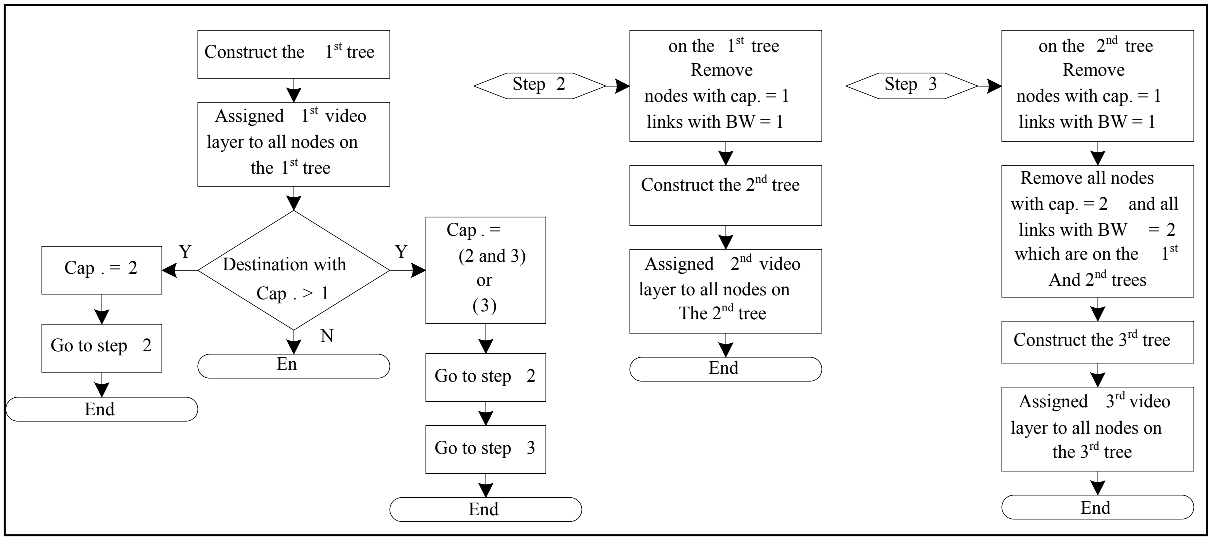

Construct the 1st tree
Assigned 1st video layer to all nodes on the 1st tree
Destination with Cap . > 1
Y
Y
N
Cap . = 2
Go to step 2
End
En
Cap . = (2 and 3) or (3)
Go to step 2
Go to step 3
End
Step 2
on the 1st tree Remove nodes with cap. = 1 links with BW = 1
Construct the 2nd tree
Assigned 2nd video layer to all nodes on The 2nd tree
End
Step 3
on the 2nd tree Remove nodes with cap. = 1 links with BW = 1
Remove all nodes with cap. = 2 and all links with BW = 2 which are on the 1st And 2nd trees
Construct the 3rd tree
Assigned 3rd video layer to all nodes on the 3rd tree
End

Figure 3.31 Construction of Hybrid-II multicast trees.

In Figure 3.32 and Figure 3.33 we plot user satisfaction and number of forwarders nodes, respectively, versus the number of destinations. The number of nodes in the network is set to be 50 and 100 nodes. As in multiple node-disjoint trees and Hybrid-I multicast trees, user satisfaction decreases as the number of destinations decreases from 5 to 25 destinations. As the number of nodes in the network increases from 50 to 100 nodes, user satisfaction increases, as shown in Figure 3.32.

Figure 3.33 shows that as the number of destinations increases, the number of forwarders nodes increases. On the other hand, as the number of nodes decreases from 100 to 50 nodes, the number of forwarders nodes decreases.

Figure 3.34 shows that as the number of nodes in the network increases, user satisfaction increases. This is because the available network resources are increased. When the number of destinations increases from 10 to 30 destinations, user satisfaction decreases, as depicted in Figure 3.34.

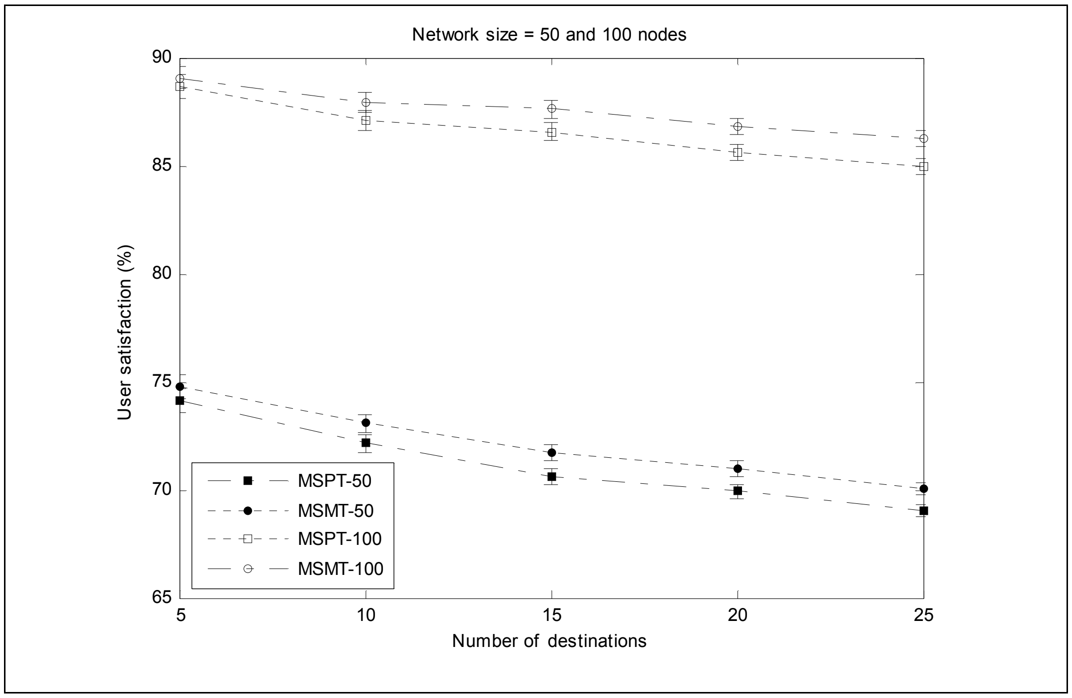


Figure 3.32 Hybrid-II multicast trees:
User satisfaction vs. Number of destinations.

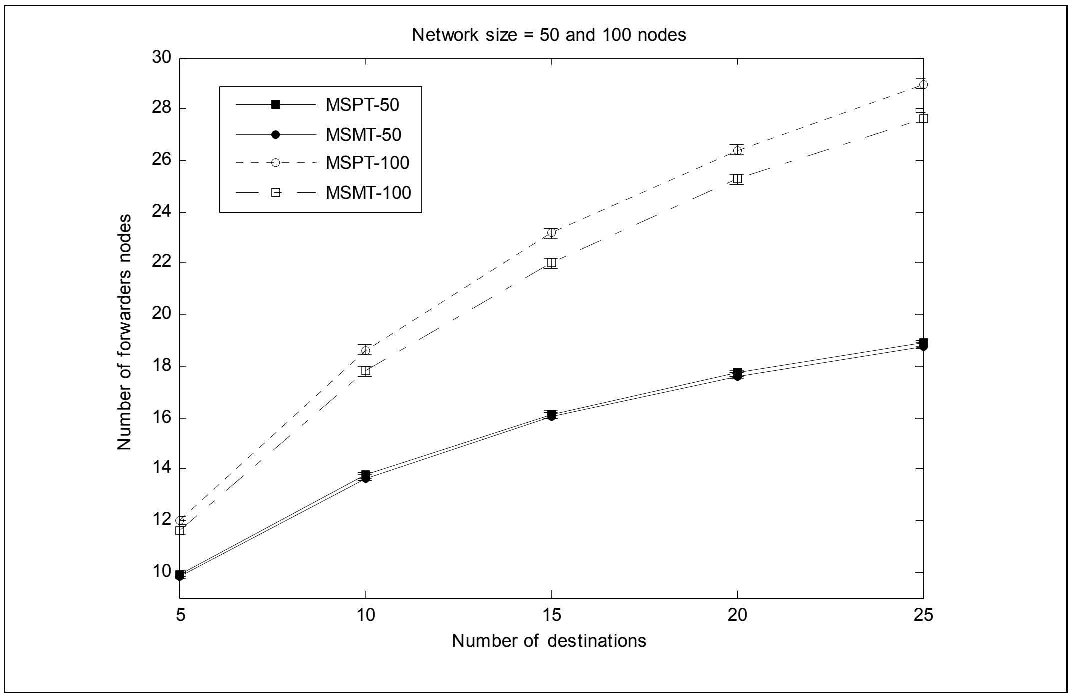


Figure 3.33 Hybrid-II multicast trees:
Number of forwarders nodes vs. Number of destinations.

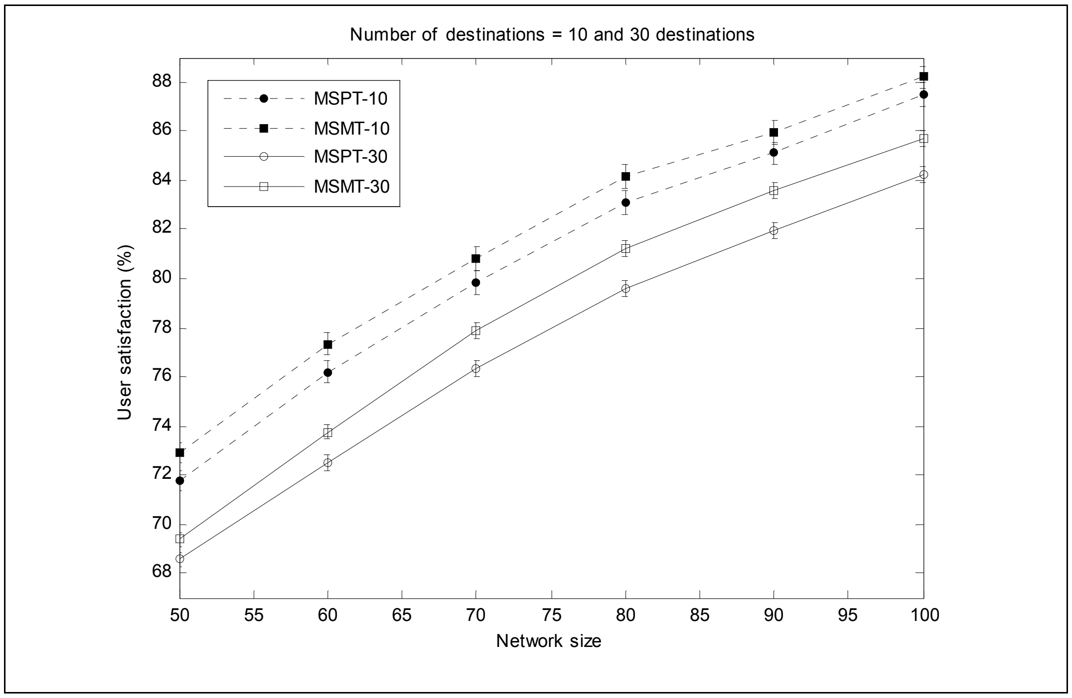


Figure 3.34 Hybrid-II multicast trees:
User satisfaction vs. Network size.

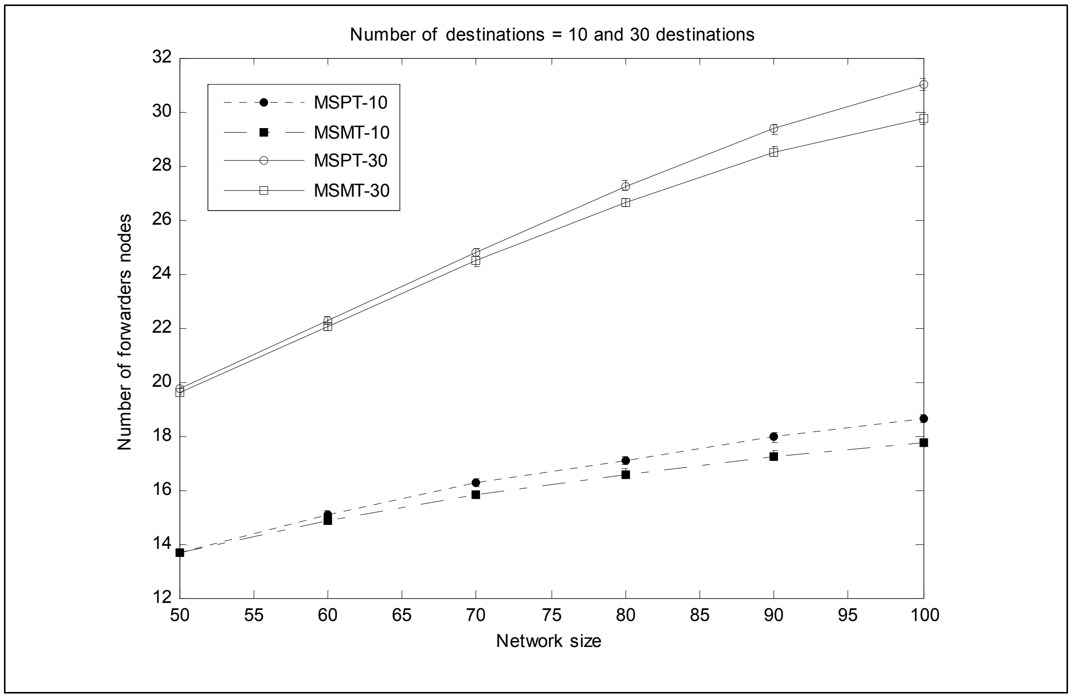


Figure 3.35 Hybrid-II multicast trees:
Number of forwarders nodes vs. Network size.

3.7.8 Multiple Node-Disjoint vs. Hybrid-I vs. Hybrid-II Multicast Trees

Simulation results demonstrate that, for both algorithms MSPT and MSMT, multiple Hybrid-II multicast trees offers higher user satisfaction than multiple node-disjoint and Hybrid-I multicast trees. This is because nodes and links are allowed to handle more than one description depending on their capacities and available bandwidth, respectively. On the other hand, in multiple node-disjoint multicast trees links and node are allowed to handle only one description. In multiple Hybrid-I multicast trees, nodes are allowed to handle more than one description according to their capacities but links can handle only one description. Clearly, multiple node-disjoint multicast trees have the lowest satisfaction because each node is allowed to handle only one video description and the number of discovered disjoint paths for each destination is small. (See Figure 3.36 - Figure 3.43). The cost for that is the robustness against links failure. In multiple Hybrid-II multicast trees, a single node failure may cause multiple paths to fail and therefore multiple destinations could be affected.

Figure 3.36 and Figure 3.37 show the user satisfaction versus number of destinations for all types of MSPT. The network size is set to be 50 and 100 nodes. Hybrid-II has higher user satisfaction than node-disjoint and Hybrid-I. This is because we allow nodes and links to be on different trees at the same time, therefore the number of available resources in the network is higher than that of link disjoint trees and node disjoint trees.

Figure 3.38 and Figure 3.39 show user satisfaction versus the network size for all types of MSPT. The number of destinations is set to 10 and 30 nodes. User satisfaction increases as the number of nodes increases. Hybrid-II offers higher user satisfaction.

In Figure 3.40 to Figure 3.43 we plot the user satisfaction of all types of MSMT versus the number of destinations and network size. To avoid repetition, the same conclusions made for all types of MSPT are applicable for all types of MSMT.

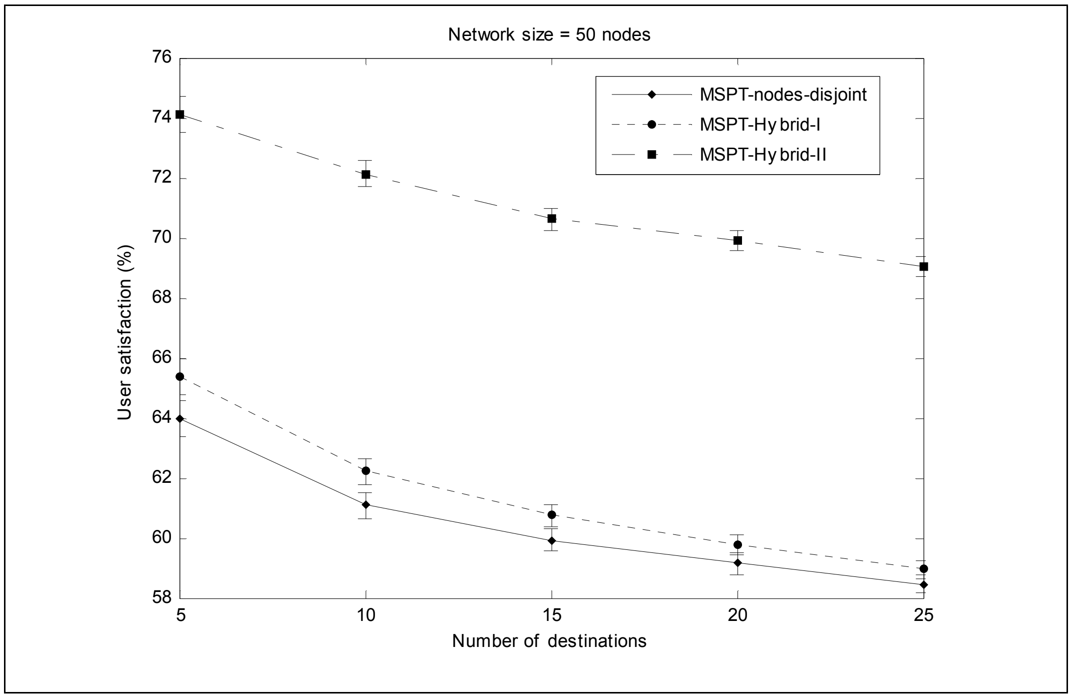


Figure 3.36 All types of MSPT: User satisfaction vs. Number of destinations. Network size = 50 nodes.

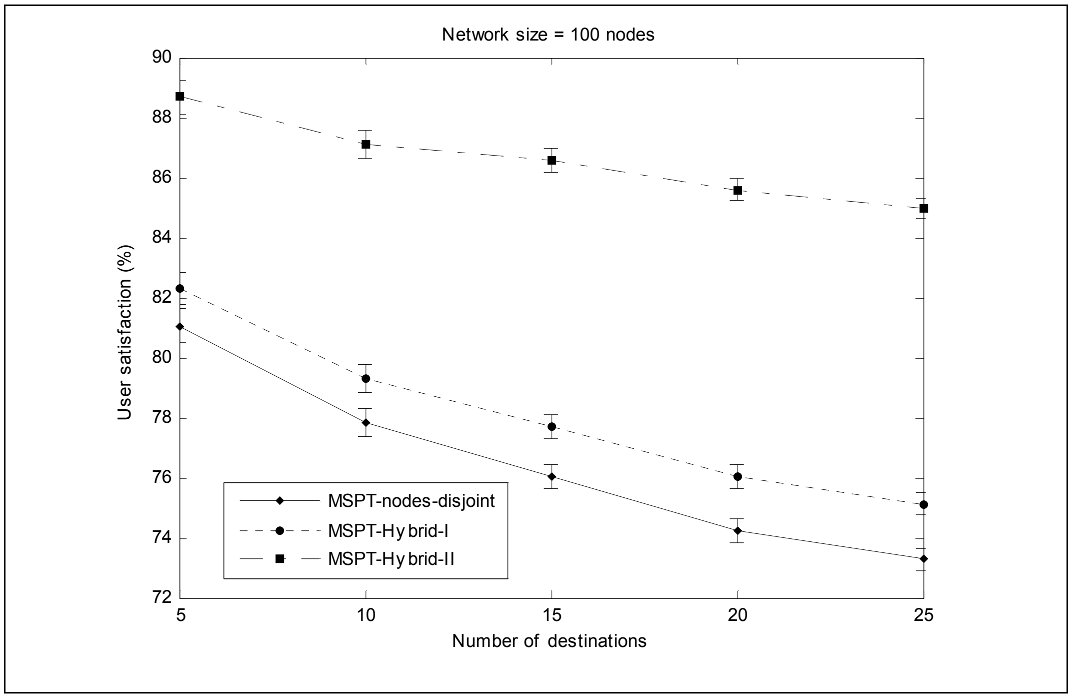


Figure 3.37 All types of MSPT: User satisfaction vs. Number of destinations. Network size = 100 nodes.

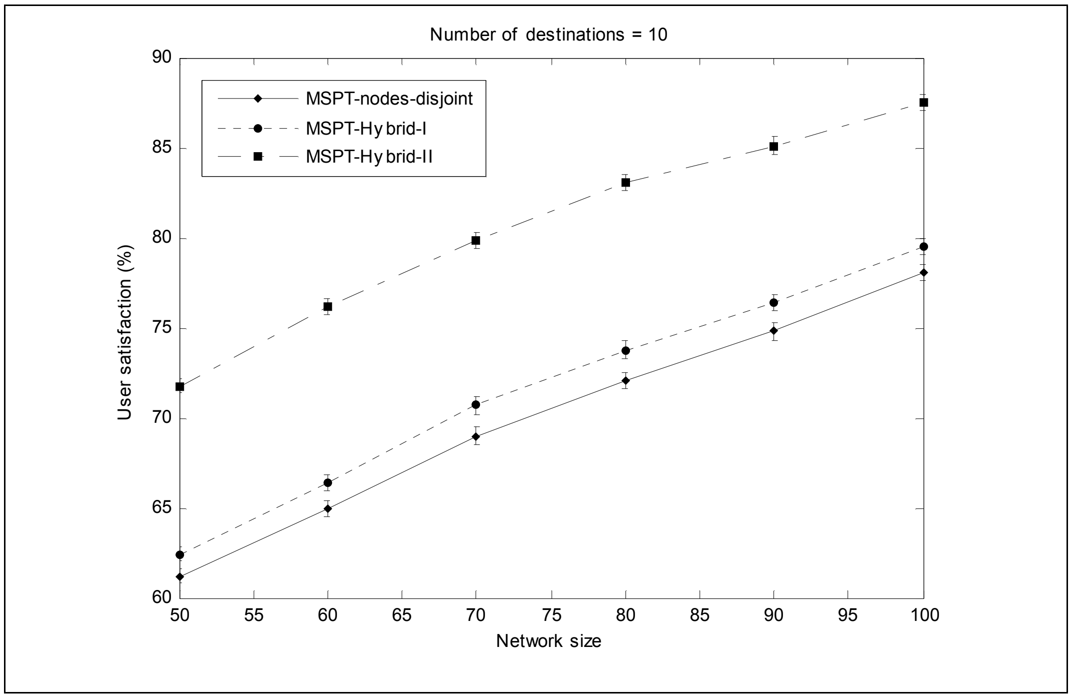


Figure 3.38 All types of MSPT: User satisfaction vs. Network size. Number of destinations = 10 nodes.

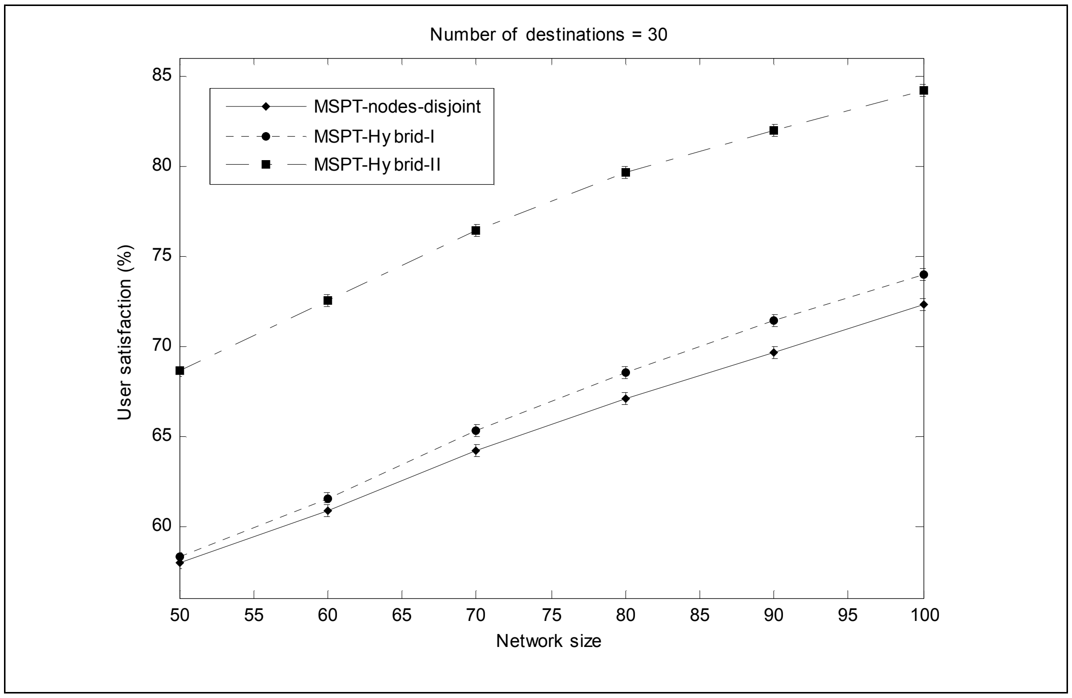


Figure 3.39 All types of MSPT: User satisfaction vs. Network size. Number of destinations = 30 nodes.

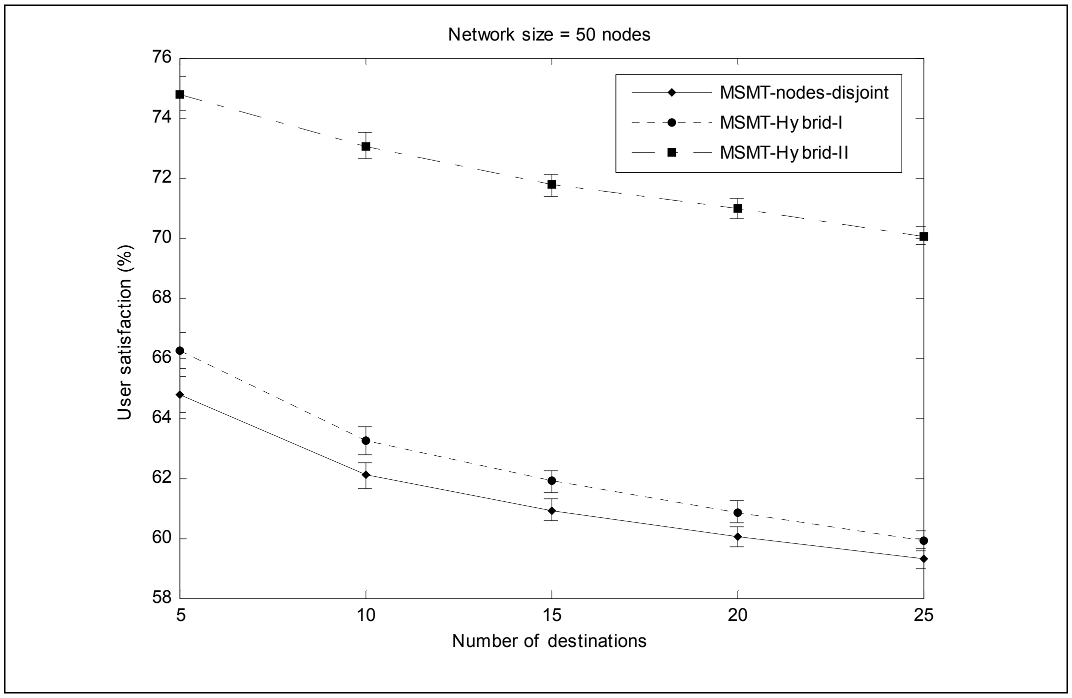


Figure 3.40 All types of MSMT: User satisfaction vs. Number of destinations. Network size = 50 nodes.

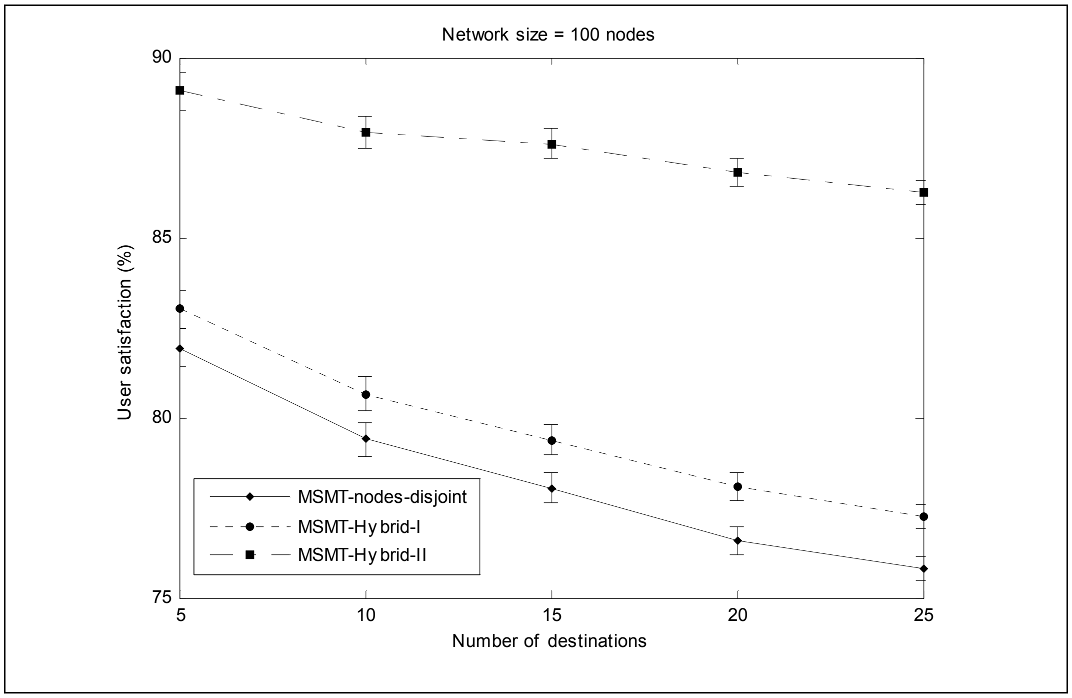


Figure 3.41 All types of MSMT: User satisfaction vs. Number of destinations. Network size = 100 nodes.

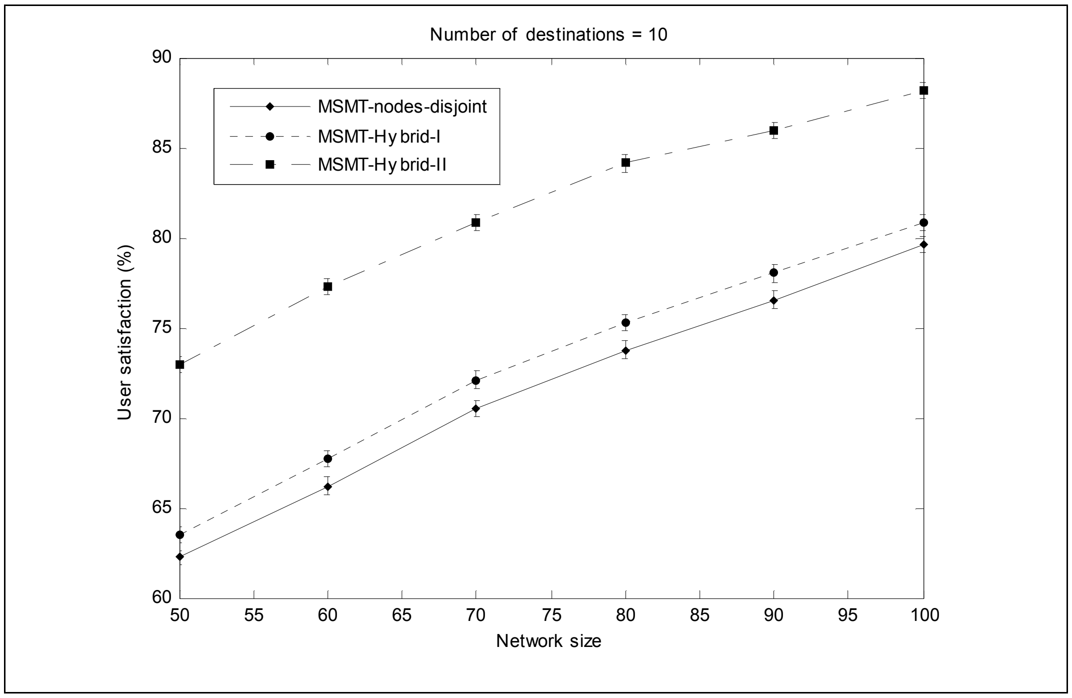


Figure 3.42 All types of MSMT: User satisfaction vs. Network size. Number of destinations = 10 nodes.

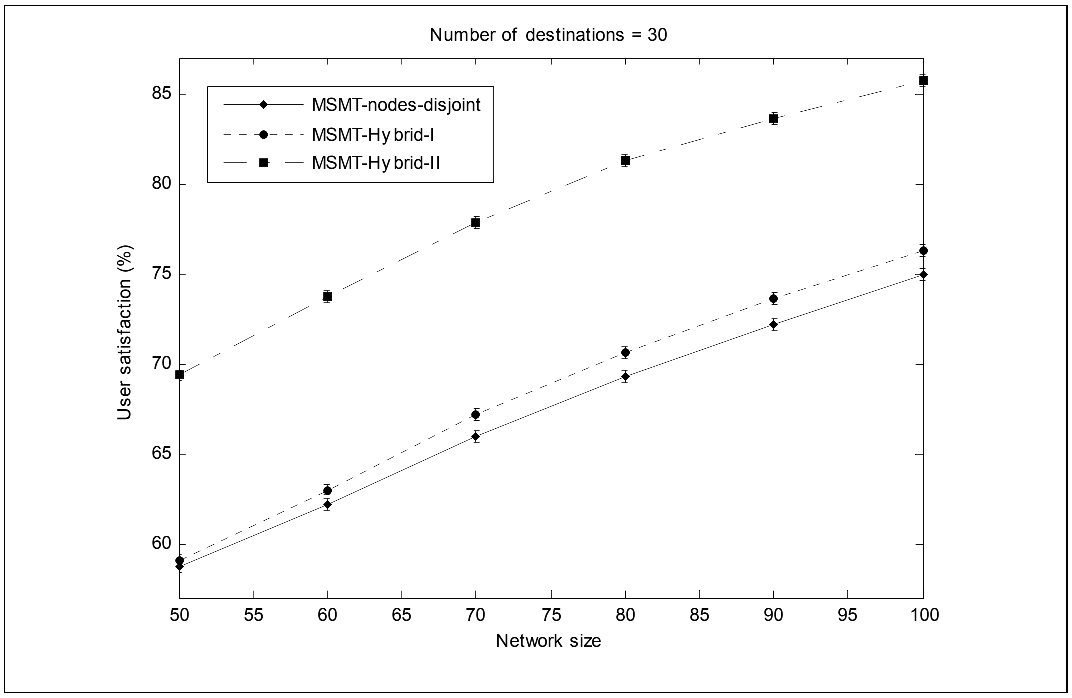


Figure 3.43 All types of MSMT: User satisfaction vs. Network size. Number of destinations = 30 nodes.

3.7.9 Multiple Multicast Trees versus Single Multicast Tree

Single multicast tree that meets the destinations' requirements (the number of video descriptions required) may not exist, even thought there are enough resources in the network to support the destinations' requirements. In this section we show that multiple multicast trees achieve higher user's satisfaction compared to single multicast tree (Figure 3.44, Figure 3.46, and Figure 3.48). The cost for that is the increase in the number of forwarders nodes (Figure 3.45, Figure 3.47, and Figure 3.49). From previous results we can see that multiple node-disjoint trees achieve lower user's satisfaction compared to Hybrid-I and Hybrid-II multicast trees. Therefore, we plot only the user's satisfaction of multiple node-disjoint trees. We can see from Figure 3.44 and Figure 3.46 that the user satisfaction of single multicast trees stays unchanged when the network size increases from 50 to 100 nodes. On the other hand, the user satisfaction of multiple node-disjoint multicast trees greatly increases.

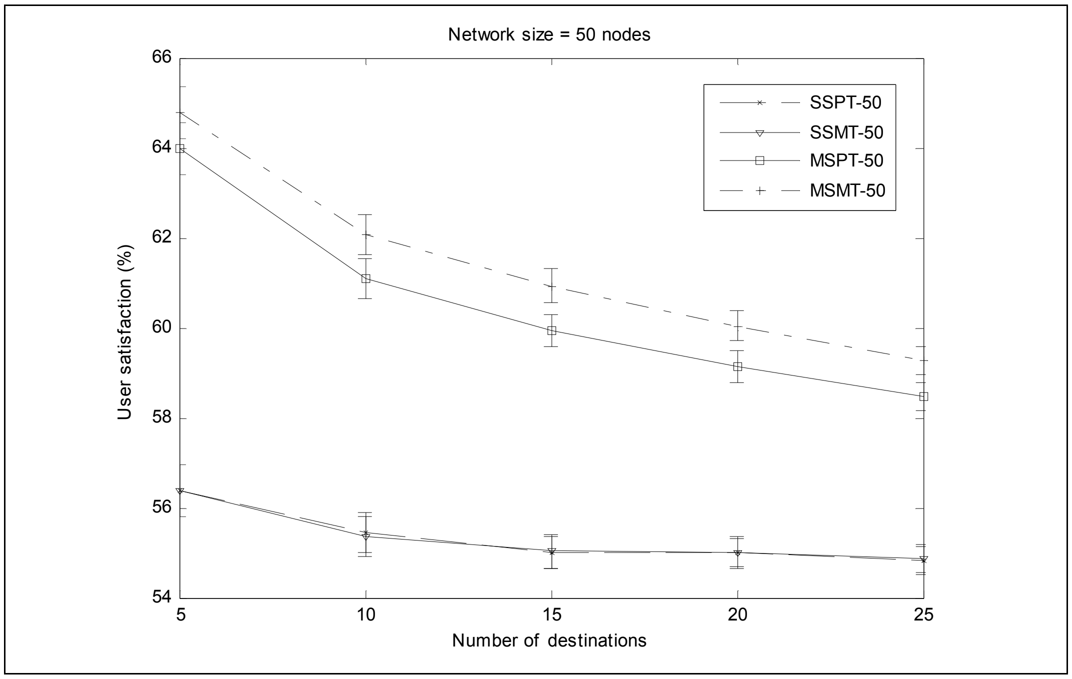


Figure 3.44 Multiple node-disjoint trees vs. Single multicast tree: User satisfaction vs. Number of destinations. Network size = 50 nodes.

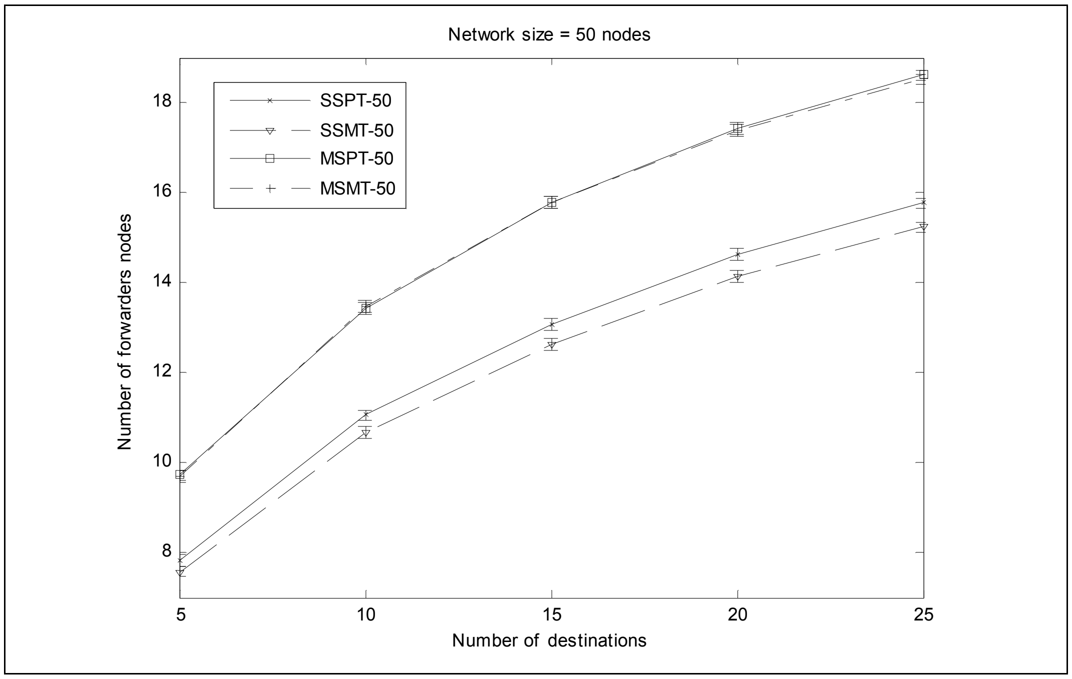


Figure 3.45 Multiple node-disjoint trees vs. Single multicast tree: Number of forwarders nodes vs. Number of destinations. Network size = 50 nodes.

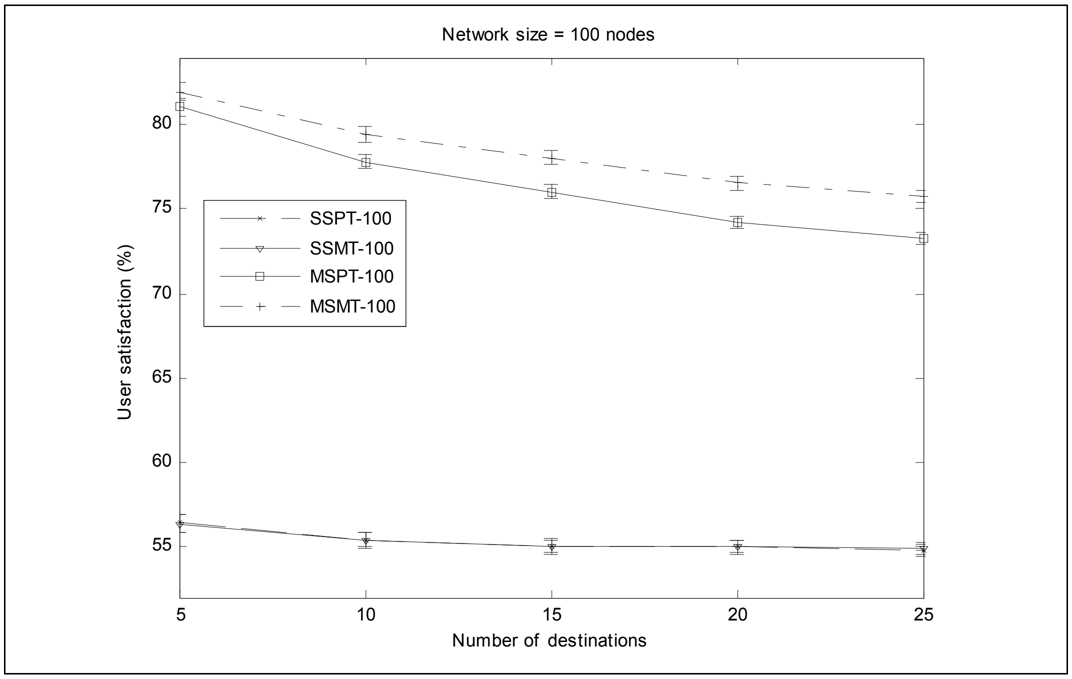


Figure 3.46 Multiple node-disjoint trees vs. Single multicast tree: User satisfaction vs. Number of destinations. Network size = 100 nodes.

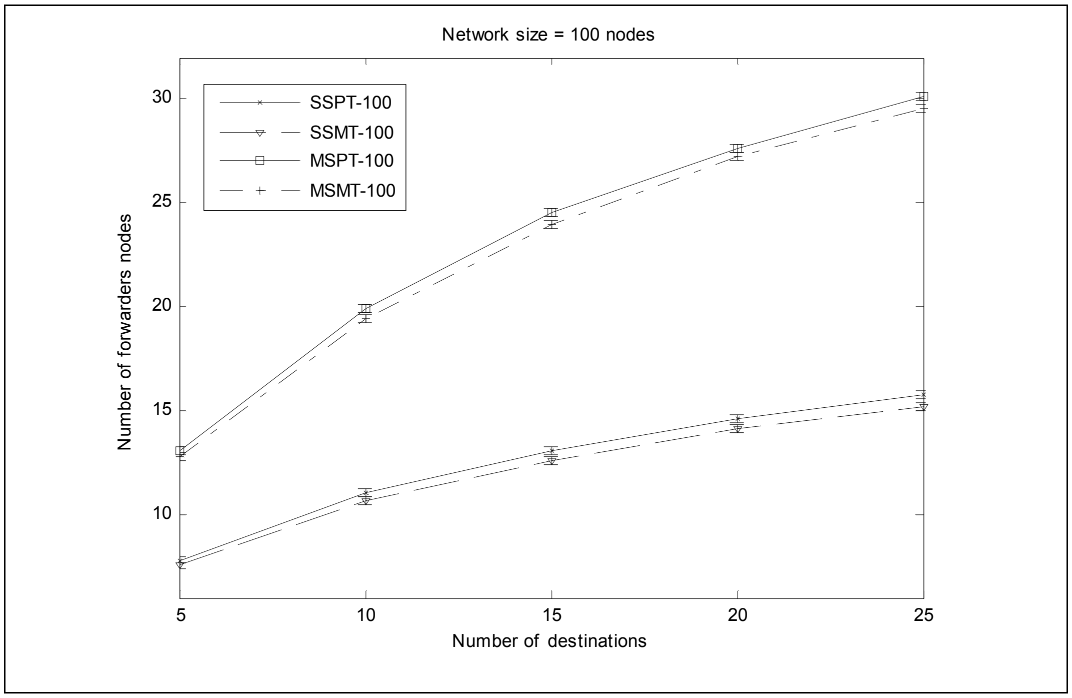


Figure 3.47 Multiple node-disjoint trees vs. Single multicast tree: Number of forwarders nodes vs. Number of destinations. Network size = 100 nodes.

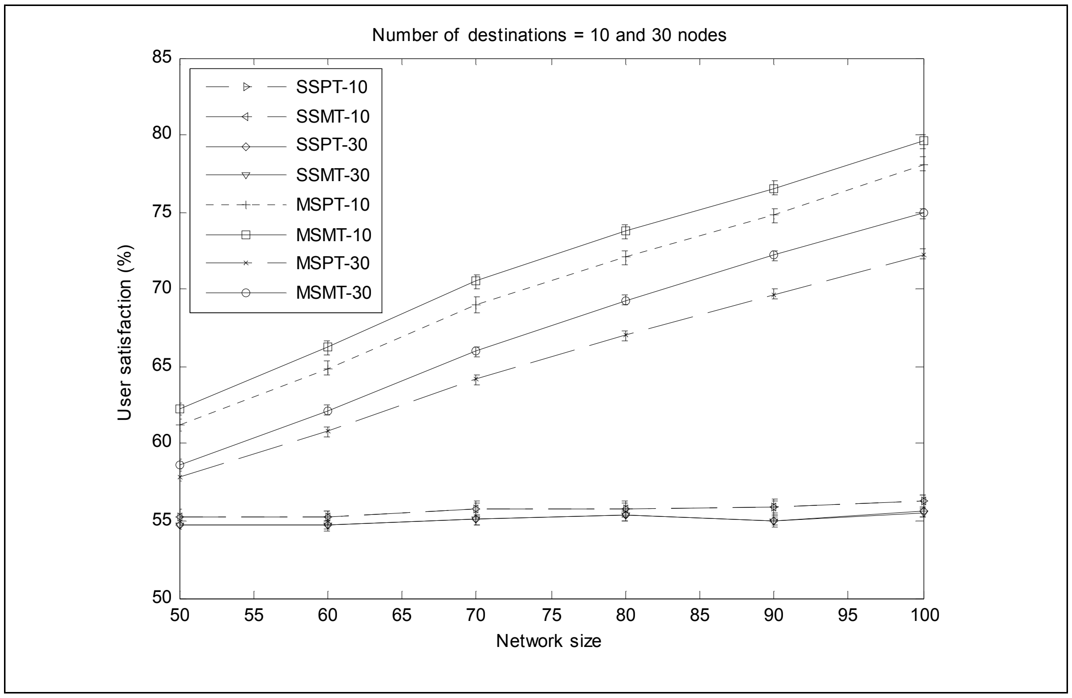


Figure 3.48 Multiple node-disjoint trees vs. Single multicast tree: User satisfaction vs. Network size.

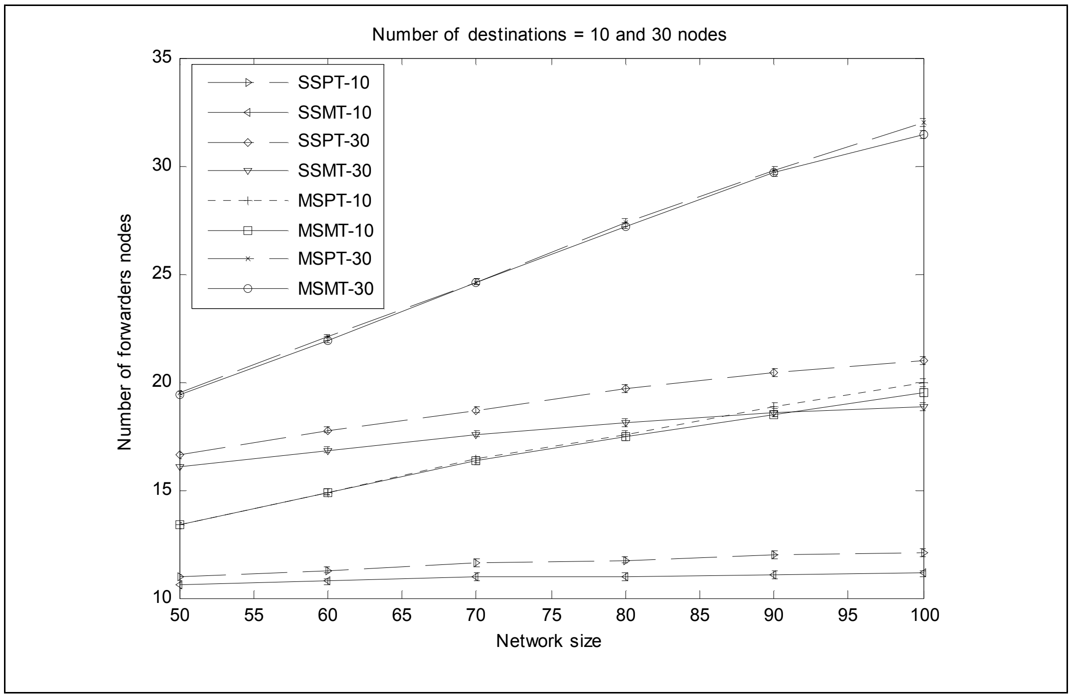


Figure 3.49 Multiple node-disjoint trees vs. Single multicast tree: Number of forwarders nodes vs. Network size.

3.8 Complexity Analysis of the Algorithms

We analyze the complexity of our proposed algorithms as follows. For Serial MDC, the shortest path algorithm (Dijkstra's algorithm) is of complexity $O(|V|\log|V| + |E|) \leq O(|V|^2)$ where $|V|$ and $|E|$ are the number of nodes and number of wireless communication links in the partial topology, respectively. Since it iterates $|\mathrm{Y}|$ times, where $|\mathrm{Y}|$ is the number of destination nodes. Therefore the complexity is $O(|V|^2 \times |\mathrm{Y}|)$ and finally the algorithm iterates $|\mathrm{N}_{req}(VD)|$ times, where $|\mathrm{N}_{req}(VD)|$ is the total number of required video descriptions for all destinations. As a result, the complexity of Serial MDC is given by $O(|V|^2 \times |\mathrm{Y}| \times \mathrm{N}_{req}(VD))$. For Distributed MDC the complexity is given by $O(|V|)$. Centralized MDC and MSPT algorithms have the same complexity of Serial MDC. Finally, For MSMT algorithms, the complexity of the Steiner tree algorithm is $O(|\mathrm{Z}||V|^2)$ where $|\mathrm{Z}|$ is the set of multicast group members (the source and the destination nodes). Since MSMT algorithm iterates $\left|\mathrm{N}_{req}(VD)\right|$ times, as a results its complexity is given by $O\left(|\mathbf{Z}||V|^2 \times \left|\mathbf{N}_{req}(VD)\right|\right)$.

3.9 Conclusion

In this chapter we study the problem of multiple multicast trees construction and the assignment of MD video. Different algorithms are proposed for that purpose. These algorithms are: Serial MDC, Distributed MDC, Centralized MDC, MSPT MDC and MSMT MDC algorithms. Some of these algorithms deploy the *independent-description* property of MDC (e.g., Serial MDC, Distributed MDC, and Centralized MDC) and the others do not deploy this property (e.g., MSPT MDC, and MSMT MCD).

Simulation results demonstrate that deploying the *independent-description* property of MDC along with multiple multicast trees can greatly improve the user satisfaction when compared to LC along with multiple multicast trees. In addition, simulation results demonstrate that the way of multiple multicast trees construction and the assignment of MD video can affect the user satisfaction. Furthermore, we show that multiple multicast trees can greatly improve the user satisfaction compared to single multicast tree. The cost for that is the increase in the number of pure forwarders nodes and the bandwidth utilization. Moreover, we show that Hybrid-II multicast trees achieve a higher user satisfaction compared to Hybrid-I multicast trees and node-disjoint trees.

CHAPITRE 4

VIDEO MULTICAST ROUTING BASED MULTIPLE DESCRIPTION CODING

4.1 Introduction

In this chapter, we propose four multicast routing protocols for video multicast over wireless ad hoc networks. In the previous chapter we showed that MDC can achieve higher user's satisfaction when compared to LC. As a result, the quality of the received video is improved. Hence, we adopt MDC as a video coding technique.

These protocols deploy the algorithms proposed in chapter 3 to construct multiple node-disjoint multicast trees and to assign MD video. The protocols are: Centralized MDMTR protocol, Sequential MDMTR protocol which is a variant of centralized MDMTR, Distributed MDMTR protocol, and Neighbor-aware MDMTR protocol which is a variant of Distributed MDMTR.

Centralized MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR deploy the *independent-description* property of MDC along with multiple node-disjoint multicast trees to improve user's satisfaction. On the other hand, Sequential MDMTR does not consider the *independent-description* property of MDC. Instead, it assigns the video descriptions in a sequential manner.

This chapter is organized as follows. In section 4.2 we introduce the CDMA over TDMA channel model. We introduce our video multicast routing protocols in Section 4.3. in Sections 4.4, 4.5, 4.6, and 4.7 we describe in details the operation of our proposed protocols, namely, Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR. We evaluate the performance of the proposed protocols in Section. 4.8. And finally, Section 4.9 concludes the chapter.

4.2 CDMA over TDMA MAC Sub-Layer and Timeslot Assignment

4.2.1 CDMA over TDMA MAC Sub-Layer

In the spread spectrum communications, a transmitter spreads the information signal in a wide frequency band by using a spreading code. A receiver uses the same code to despread the received signal and retrieve the data. This approach provides multiple access capability, named CDMA, by allowing the simultaneous transmission of packets by different nodes (Fantacci, Ferri et Tarchi, 2005). A defining characteristic of CDMA is the possibility of receiving multiple packets at the same time. Spreading code schemes can take several different forms in CDMA wireless ad hoc networks, including network-wide code where a common code is used by all users, receiver-based code and transmitter-based code. In CDMA wireless ad hoc networks, when multiple transmitters transmit to the same receivers at the same time, packet collisions occur (Zhou, Li et Yang, 2005). CDMA, which integrates spread spectrum technique with multichannel mechanism, may provide a solution for the above packet collision. Recently, many multiple access protocols based on CDMA are proposed for ad hoc networks, which can simultaneously transmit multiple packets in the same frequency band by means of CDMA (Liu, Cai et Tu, 2006).

CDMA is attractive because of its features of asynchronous operation, ability to add new users, multipath rejection, and antijamming capacity. In a CDMA system, simultaneous transmissions can be isolated by using different spreading codes. However, a node in a CDMA system needs to know which code to use in transmitting or receiving a particular packet (Hu, 1993). This code assignment problem is trivial if the network size is small. It becomes very inefficient to assign a unique code to each transmitter or receiver when the network size grows. Spatial reuse of codes becomes increasingly important when we extend CDMA to a large multihop ad hoc network. Therefore, the purpose of the code assignments is to spatially reuse spreading codes to reduce the possibility of packet collisions and to react dynamically to topological changes (Hu, 1993).

Wireless ad hoc networks (due to the absence of a coordination device) arise the problem of assigning CDMA codes to the nodes and to the communications between them. In the literature there are several code assignment methods for wireless ad hoc networks. Among them, four basic types of code assignment (Hung, Law et Leon-Garcia, 2002) exist:

- Common Code Assignment (CCA): All the nodes have a common code (Sousa et Silvester, 1988). The addressing information is placed inside the packet header in order to identify the source and destination. All the nodes monitor the same common code for any packet arrival. Multicast and broadcast communications can be implemented.

- Receiver-based Code Assignment (RCA): Each node has its own receiver-based code (Sousa et Silvester, 1988). The transmitter has to look up a code assignment table to find out the code of its receiver and then send the data packet on the receiver-based code. The receiver has to monitor its code all the time for any packet arrival. In this scheme, a receiver only has to listen to one code, but primary conflicts cart occur. The RCA problem is to find a spreading code for each node to use in receiving packets with the constraint that all logical neighbors of a given (transmitting) node have different receiving codes.

- Transmitter-based Code Assignment (TCA): Each node has its own Transmitter-based Code (Makansi, 1987). The transmitter sends its data on its own code. The receiver must monitor all transmitter-based codes of its neighbors because the receiver does not know in advance which node will transmit. The TCA problem is to find a spreading code for each node to use in transmitting packet with the constraint that all logical neighbors of a given (receiving) node have different transmitting codes, i.e., a transmitter uses its unique code for transmission, and no primary conflict, where two transmission is the same code arrive at a receiver simultaneously, would result from logical neighbors' transmission. A spreading code can be reused if two nodes have a hop distance greater than two.

- Pairwise Code Assignment (PCA): Each pair of nodes has assigned a unique code (Hu, 1993). The transmitter will look up a code assignment table to find out the code to communicate with a specific receiver. The receiver must monitor a set of codes simultaneously.

RCA schemes are cheaper and simpler, but yield a lower throughput than TCA ones. RCA schemes retain the same interference avoidance properties of TCA ones, while requiring a more expensive hardware and (in some cases) smaller number of codes, and yielding a slightly worse performance (Bonuccelli, 1995). Moreover, hidden terminal interferences cannot be completely avoided by RCA schemes, while they can be totally eliminated by properly assigning orthogonal codes in TCA schemes. In PCA, a very large number of orthogonal codes are needed in a fully connected network. However, this scheme can result in a smaller number of codes than RCA or TCA scheme in a carefully controlled topology.

For multimedia transmission (voice or video), the connection-oriented approach is usually taken. Such connections do not permit interferences or competitions from others in order to guarantee the delay and jitter. For such a high QoS guarantee, the multimedia transmission needs an independent channel with its exclusive bandwidth. In a pure CDMA system, a single code needs to be assigned to a multimedia transmission at the least, even though a code may provide much more bandwidth than the connection requires. When there are multiple simultaneous connections, either we use more codes (and hence more hardware) to support multiple connections or we can use the TDMA approach on top of the CDMA mechanism to multiplex connections that requires only a portion of the code bandwidth. This resource efficiency issue becomes more significant as the multihop ad hoc network becomes larger. When the time is divided into multiple time slots, more simultaneous (sub-rate) connections can be served in a time efficient manner. Each connection can independently occupy one time slot until the end of the connection. A TDMA system is advantageous from the viewpoint that the number of radio transceivers can be decreased, and that system control functions, such as hand-off control, can be easily implemented. Monitoring for other channels before hand-off is performed at time slots that are not dedicated for the current communications.

We thus consider the well-known CDMA over TDMA channel model at the MAC sub-layer proposed in (Lin et Liu, 1999) (Lin, 2001). That is, CDMA is overlaid on top of the TDMA infrastructure. Namely, multiple sessions can share a common TDMA slot via CDMA. The transmission and reception between two neighbours at the MAC layer are governed by the TDMA model. A node that wishes to transmit signals must use a free timeslot for transmission, and the node that wishes to receive the signals needs to listen to the transmitting node in the same timeslot. In this channel model, a radio station can only receive a single transmission at a time and cannot transmit and receive simultaneously. In addition, the channel is assumed to be time slotted. All nodes keep accurate common time (there exits a global clock or time synchronization mechanism). Each slot includes the number of redundant bits (e.g., for error control coding, retransmission) that must be sent when the channel has a low signal-to-noise ratio in order to get a successful transmission. Furthermore, the assumptions found in most other radio data link protocols (Baker et Ephremides, 1981; Chlamtac et Kutten, 1985; Chlamtac et Pinter, 1987a; Gerla et Tsai, 1995; Goodman et al., 1989; Lin et Gerla, 1997) are considered. That is, the physical layer can provide the service of the slotted channel. Only one data packet can be transmitted in each data slot.

Consider the example of the wireless network shown in Figure 4.1 that uses TDMA for data transmission. The mobile host *A* intends to transfer data to the mobile host *D*. All slots in the TDMA frame are assumed to be free. Suppose that *A* reserves slots 1 and 2 for transmitting data to *B*, and *B* uses slots 3 and 4 to forward packets to *C*. Because *A* and *C* are hidden from each other, *C* may want to send packets to *D* by using slots 1 and 2. Thus, there will exist a collision at *B* (because *B* may receive packets from *A* and *C* simultaneously) (Chlamtac et Pinter, 1987b).

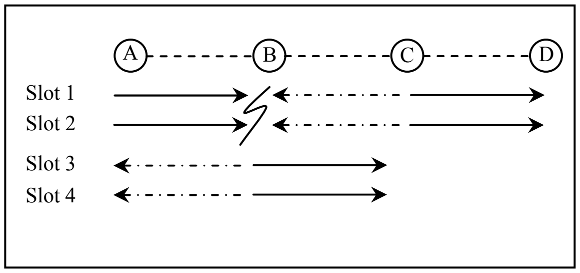


Figure 4.1 Hidden terminal problem.
Taken from Lin (1999, p.1427)

CDMA can be used to solve this problem (all spreading codes are assumed to be orthogonal to each other). For example, consider the same topology in Figure 4.1. Figure 4.2 shows that when *B* uses slots 3, 4 to transmit packets to *C*, we can assign a code (say code 2) to node *B* which is different from the code (say code 1) used by *A*. That is, we use a transmitter-based code assignment to assign a code to each transmitter for data transmission. A spreading code can be reused if two nodes have a hop distance greater than two (Chlamtac et Pinter, 1987b). In Figure 4.2, *C* can use the same slots (1 and 2) as *A* to send packets to *D* encoded by a different code (say code 3) without any collision at *B*. It is notable that this case is assumed to be only one session through *A*, *B*, *C*, and *D*. If *A* and *C* (different sessions) intend to send packets to *B* in the same slot, then only one packet can be received and another will be lost depending on which code *B* locks on.

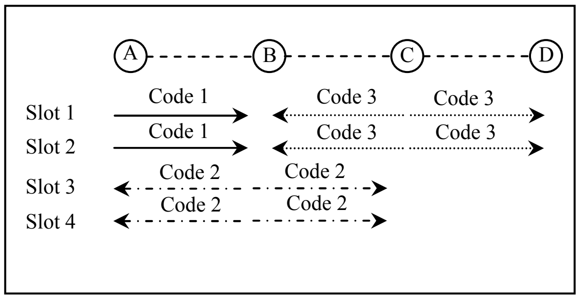


Figure 4.2 CDMA over TDMA.
Taken from Lin (1999, p.1427)

4.2.2 Timeslot Assignment and Bandwidth Calculation

In TDMA a time frame is divided into two phases: a control phase and data phase. The size of each slot in the control phase is much smaller than the one in the data phase. The TDMA time frame structure is shown in Figure 4.3. The control phase uses pure TDMA full power transmission in a common spreading code. That is each node takes turns to broadcast its information to all of its neighbors in a predefined slot, such that the network control functions can be performed distributively. The control phase is used to perform all control functions, such as slot and frame synchronization, power measurement, code assignment, slots request, etc. We assume the information can be heard by all of its adjacent nodes. In a noisy environment, where the information may not always be heard perfectly at the adjacent nodes, an acknowledgment scheme is performed in which each node has to acknowledge for the last information in its control slot. By exploiting this approach, there may be one frame delay for the data transmission after issuing the data slot reservation.

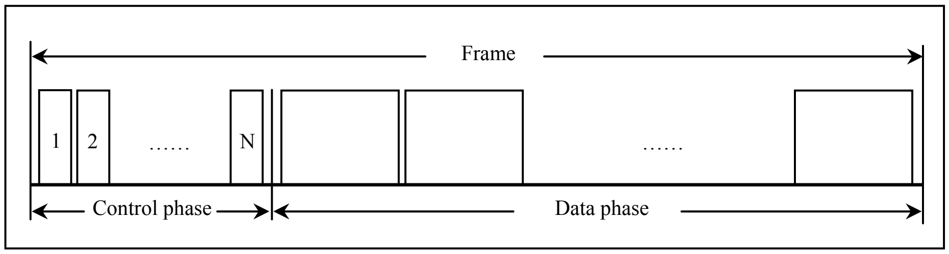


Figure 4.3 TDMA frame structure.
Taken from Lin (1999, p.1428)

Ideally, at the end of the control phase, each node has learned the channel reservation status of the data phase. This information will help one to schedule free slots, verify the failure of reserved slots, and drop expired real-time packets.

Because only adjacent nodes can hear the reservation information and the network is multihop, the free slots recorded at every node may be different. The number of free timeslots over a link represents the free bandwidth on the link. The bandwidth is measured in the unit of free timeslots. For example in Figure 4.4(a), the free bandwidth of link (*A*, *B*) is 2 because it has 2 free timeslots. Notice that, the free bandwidth over a path depends not only on the free timeslots over the links in the path, but also on the slot assignment method. Sometimes, even though every link has a free timeslot, the path does not have one unit bandwidth. For example in Figure 4.4 (a), the slot assignment on link (*B*, *C*) conflicts with the assignment on link (*C*, *D*), because slot 5 was assigned to both links and node *C* cannot do both receiving from *B* and transmitting to *D* simultaneously at slot 5. Figure 4.4(b) shows a good case of slot assignment, which provides one unit of bandwidth for the path.

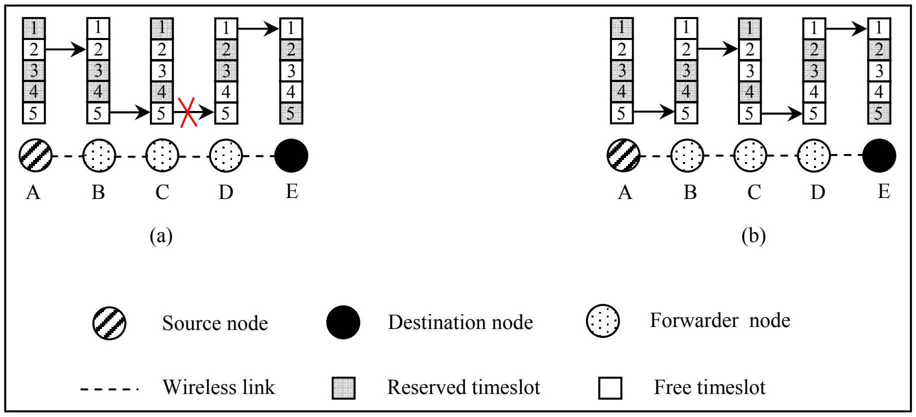


Figure 4.4 Examples of timeslot assignment.

Assigning free timeslots to a path to maximize the available bandwidth of the path is NP-hard (Lin et Liu, 1999). In this work, the timeslot assignment algorithm follows the method proposed in (Lin et Liu, 1999).

The *path bandwidth* (which can be called *end-to-end bandwidth*) between two nodes, that are not necessarily adjacent, to be the set of available slots between them. If two nodes are adjacent, the path bandwidth is the link bandwidth. Consider the example in Figure 4.5, and assume that one hop distance is between *A* and *B*. If *C* has free slots {1, 3, 4}, and *B* has free slots {1, 2, 3}. Then the *link bandwidth* between *C* and *B* is {1, 3}. This means that we can only exploit slots 1 and 3 for packet transmission from *C* to *B*. Thus, if a session needs more than two slots in a time frame, then it will be rejected to pass through (C, B).

The *link bandwidth* between two adjacent nodes, say *A* and *B*, is defined as:

$$linkBW = free\_slot(A) \cap free\_slot(B)$$

where $free\_slot(X)$ is the slots which are not used by any adjacent host of *X* to receive or to send packets from the point of view at node *X*.

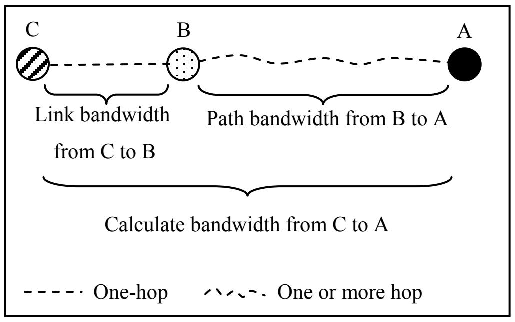

C
B
A
Link bandwidth from C to B
Path bandwidth from B to A
Calculate bandwidth from C to A
One-hop
One or more hop

Figure 4.5 Bandwidth information calculation overview.
Taken from Lin (1999, p.1429)

Further, the *link bandwidth* can be deployed to compute end-to-end bandwidth. End-to-end bandwidth can provide us with an indication of whether there exits a QoS route between a given source-destination pair. The following four cases are used as examples to show how to calculate the path bandwidth.

- **Case 1:** Assume the link bandwidth of both (*A*, *B*) and (*B*, *C*) are the same, say {1, 2, 3, 4}, as in Figure 4.6. If *C* uses slots 1, 2 to send packets to *B*, then *B* can only use slots 3, 4 to forward packets to *A*. This is because *B* cannot be in transmitting mode and listening mode simultaneously. So the path bandwidth from *C* to *A*, denoted as *path*BW(*C*, *A*), can be {1, 2}, and its size is two. In this case, four free slots can only contribute two slots for path bandwidth. Namely, [4/2] = 2. Similarly, if there are only three free slots on both links, then the size of path bandwidth is [3/2] = 1.

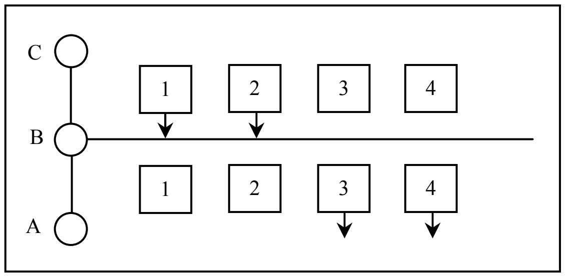


Figure 4.6 Equal case.
Taken from Lin (1999, p.1429)

- **Case 2:** Assume *link*BW(*A*, *B*) = {2, 3} and *link*BW(*B*,*C*) = {1, 2, 3, 4}, as in Figure 4.7. Namely, *link*BW(*A*, *B*) $\subset$ *link*BW(*B*, *C*). If *C* uses slot 2, then *B* cannot use slot 2 any more. So in this case, *C* should first use slots in *link*BW(*B*, *C*) – *link*BW(*A*, *B*) = {1, 4} to maximize system utilization. Therefore, if *C* uses slots 1, 4, then *B* can use slots 2, 3. So *path*BW(*C*, *A*) = {1, 4}, and its size is two. Similarly, we can use the same way to process the case of *link*BW(*A*, *B*) $\subset$ *link*BW(*B*, *C*). In this case, *B* must use slots in "*link*BW(*A*, *B*) – *link*BW(*B*, *C*)" first.

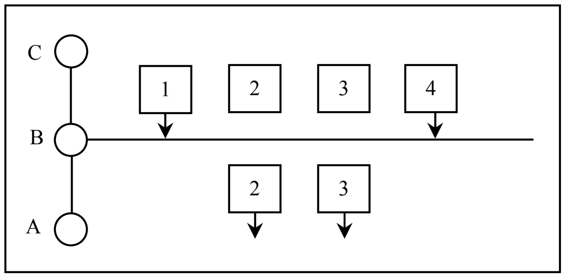


Figure 4.7 Containing case.
Taken from Lin (1999, p.1429)

- **Case 3:** If *link*BW(*A*, *B*) $\cap$ *link*BW(*B*, *C*) = ϕ , no conflict will occur. Figure 4.8 shows this example. *C* can choose either slot 3 or 4, and *B* chooses slot 2. So *path*BW(*C*, *A*) = {3}, and its size is one.

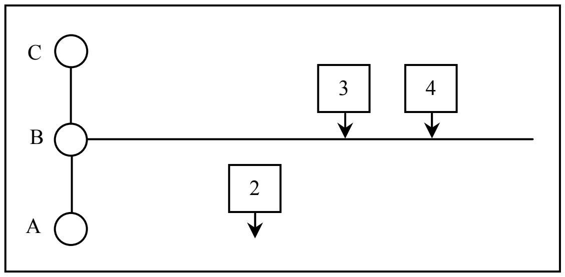


Figure 4.8 Exclusive case.
Taken from Lin (1999, p.1429)

- **Case 4:** This is a general case, as shown in Figure 4.9. We will find any general case can be regarded as a combination of the previous three cases. Follow the slot assignment policy in Case 2. We assign slot 9 to *C* and slot 1 to *B* first. Figure 4.10 shows the slots left. Next, we assign slot 10 to *C* and slot 4 to *B*. Figure 4.11 shows only slots {5, 6, 7, 8} left (slot 4 cannot be used by C any more since B is in transmitting mode). At present, this is the same situation as in Case 1. So C can be assigned slots 5, 6, and B is assigned slots 7, 8, as shown in Figure 4.12. Here we let the *path*BW(C, A) = {5, 6, 9, 10}. That is, C can use slots {5, 6, 9, 10} to send packets to B, and then B uses {1, 4, 7, 8} to forward packets to A. The size of path bandwidth from to is four. The general case is a combination of Cases 1–3. This slot assignment policy considers the slots not in *link*BW(B,C) $\cap$ *link*BW(A,B) first, until one of the special cases (i.e., Cases 1–3) occurs.

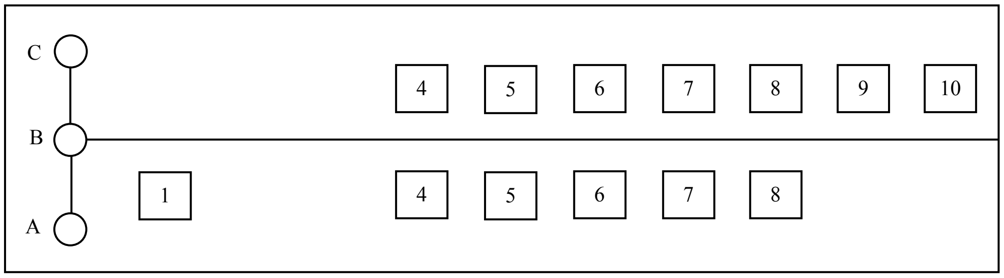


Figure 4.9 General case.
Taken from Lin (1999, p.1429)

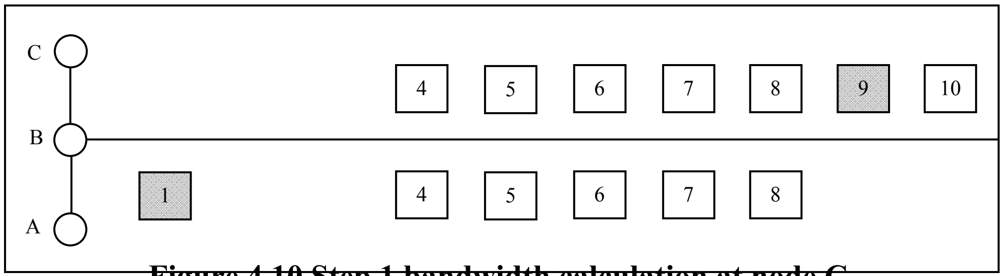


Figure 4.10 Step 1 bandwidth calculation at node C.
Taken from Lin (1999, p.1429)

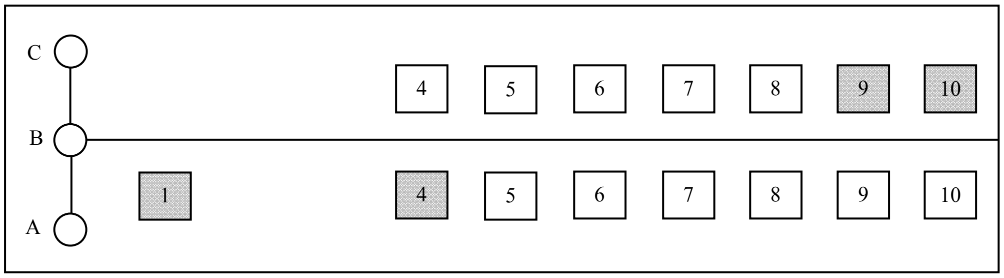


Figure 4.11 Step 2 bandwidth calculation at node C.
Taken from Lin (1999, p.1429)

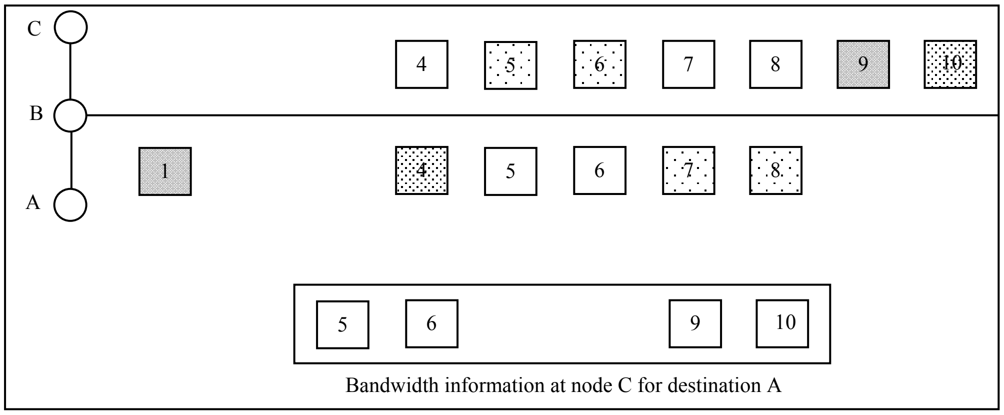


Figure 4.12 Final result of bandwidth at node C.
Taken from Lin (1999, p.1430)

We have described, above, how to calculate path bandwidth. In the following, we will discuss how to perform the slot assignment. We use the example in Figure 4.9 to describe how to do the slot assignment. For a given path, the source node, intermediate nodes, and the destination node will perform different work.

- **Source Node:** According to *link*BW(*C*, *B*) and *path*BW(*B*, *A*), we can compute the path bandwidth *path*BW(*C*, *A*) = {5, 6, 9, 10} as we mentioned previously (Figure 4.12). Thus, the node *C* can reserve any of these slots. Assume the session from *C* to *A* needs four data slots in each time frame (i.e., the QoS requirement). Thus {5, 6, 9, 10} are reserved, and we must remove these slots from the path bandwidth in the other destination entries in the routing table. For example, for the destination *E*, if *path*BW(*C*, *E*) contains slots 5, 6, 10 and the size of the path bandwidth is five, then *C* must remove slots 5, 6, 10 away, and the size becomes two (5 – 3 = 2). Figure 4.13 shows slots {5, 6, 9, 10} are reserved and then marked.

- **Intermediate Nodes:** The *path*BW(*B*, *A*) = {1, 4, 5, 6, 7, 8}. When *B* receives the reservation packet from *C*, it must check if its slots {5, 6, 9, 10} are free. If so, then it can receive data packets from *C*. Notice that *B* must remove these slots from the path bandwidth in each destination entry in the routing table. In addition, *B* must check if there are free slots which can be used for forwarding packets to next hop (i.e., *A*). In this case, slots {1, 4, 7, 8} are available. Thus, *B* reserves them (as shown in Figure 4.14). In the meantime, *B* must delete these slots from the path bandwidth field in the routing table. If any one slot in {5, 6, 9, 10} is busy or if there are no enough free slots (four slots in this case) to forward the packets, this reservation will fail. At this time, *B* must reject the reservation request. Because *C* has already reserved slots {5, 6, 9, 10}, *B* needs to send a control packet, say *RESET*, to *C* to ask to free these slots. These checking operations have to be done because the topological change may affect the free slots.

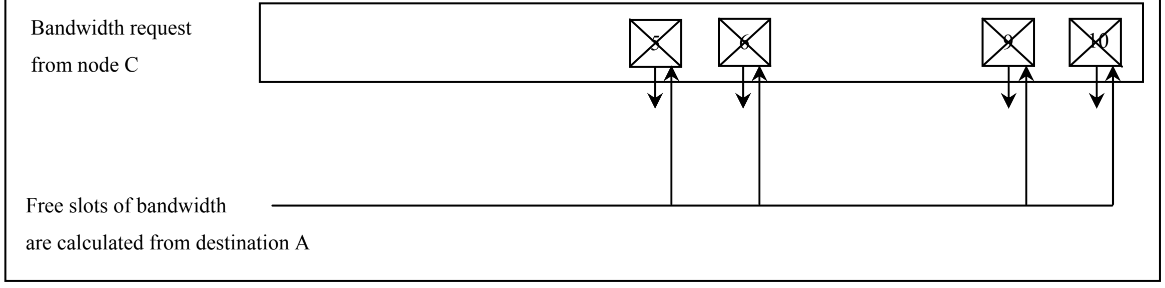


Figure 4.13 Bandwidth information at node C.
Taken from Lin (1999, p.1431)

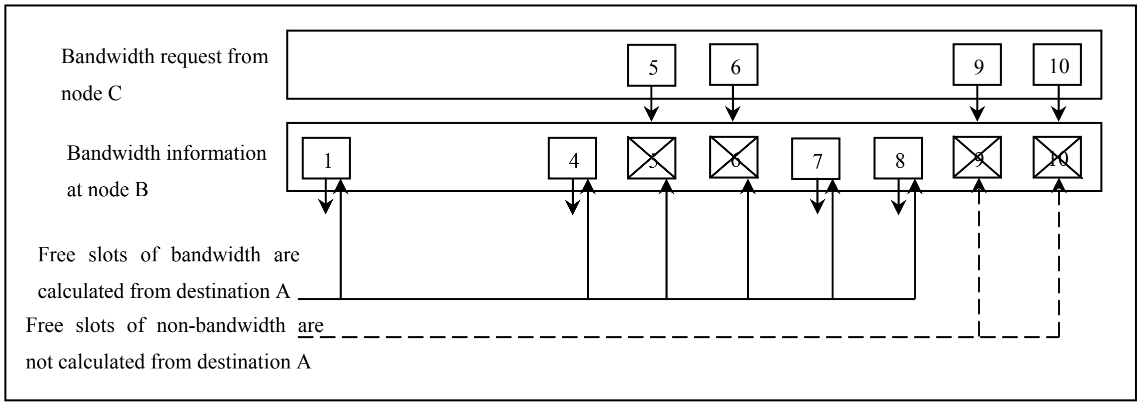

Bandwidth request from node C
5
6
9
10
Bandwidth information at node B
1
4
5
6
7
8
9
10
Free slots of bandwidth are calculated from destination A
Free slots of non-bandwidth are not calculated from destination A

Figure 4.14 Bandwidth information at intermediate node B.
Taken from Lin (1999, p.1431)

Destination Node: When the destination *A* receives the reservation packet from the previous hop (i.e., *B*), it only reserves slots {1, 4, 7, 8} for receiving data packets from *B*, as shown in Figure 4.15. These slots must be deleted from the path bandwidth field in the routing table. Like intermediate nodes, if *A* finds that any one slot in {1, 4, 7, 8} is not free, it must send the RESET message back to free those reserved slots hop-by-hop.

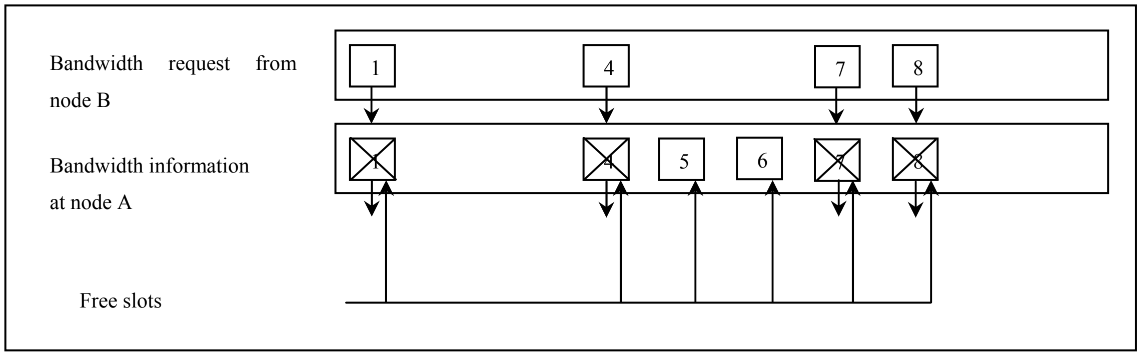

Bandwidth request from node B
1
4
7
8
Bandwidth information at node A
1
4
5
6
7
8
Free slots

Figure 4.15 Bandwidth information at node A.
Taken from Lin (1999, p.1431)

4.3 Video Multicast Routing Protocols

In the following sections, we propose four on-demand multicast routing protocols for video transmission over wireless ad hoc networks. In the first routing protocol, the assignment of MD video and the construction of multiple disjoint multicast trees are performed in a centralized way, for short Centralized MDMTR. In the second routing protocol, the assignment of MD video and the construction of multiple disjoint multicast trees are performed in a distributed way. We refer to this protocol as Distributed MDMTR. Sequential MDMTR protocol is a variant of Centralized MDMTR. However, in contrast to Centralized MDMTR, Sequential MDMTR does not consider the *independent-description* property of MDC. Instead, Sequential MDMTR assigns the video descriptions in a sequential manner. Neighbor-aware MDMTR is a variant of Distributed MDMTR protocol. In Neighbor-aware MDMTR protocol each node waits for a short time before it decides which video description it should select. The selection of video description at a given node depends on which video description its neighbor nodes have decided to select. Thus, a node can select different video description than its neighbor nodes. In order to minimize the packet drop between two trees, all the protocols construct totally node-disjoint multicast trees.

The multicast source S generates a number of MD video, say $\mathbf{M}$, where each additional description represents a *QoS_level*. For example, if there are three video descriptions available at the multicast source it represents three possible *QoS_level* (e.g., the first, second, or third description represents *QoS_level* one, any two different descriptions represent *QoS_level* two, and three different descriptions represent *QoS_level* three). In this study, we limit the number of *QoS_level* to *two*, i.e., there are two video descriptions.

4.3.1 Multicast Packet Forwarding Scheme

Multicast packet forwarding is based on the source S of the multicast packet, multicast *groupId*, and multicast *treeId*. Our proposed protocols constructs and maintains totally multiple disjoint multicast trees. Each *tree t* is used to deliver different description of MDC video concurrently. However, the multicast packet forwarding scheme does not support packet forwarding across different trees.

A traffic allocator, at the application layer, is used to split the video traffic. In addition to multicast source S address and the multicast *groupId*, each packet contains a *treeId* field to identify the tree it is meant for. The multicast packet forwarding scheme works as follows. When an intermediate node receives a data packet, it checks its *Membership table* and *Message cash* to avoid forwarding duplicate data packet. The node forwards a non-duplicate data packet (to its downstream node) to a multicast *tree t* if it's a forwarder in that tree, otherwise it drops the packet. This process continues until the packet reaches a leaf destination node.

4.3.2 Data structure

- **Multicast routing table:** Each node creates and maintains a Routing Table for the source S, and multicast *groupId*. It stores the source address, multicast group *groupId*, the addresses of the upstream and downstream node for the *tree t*, minimum hop count from the source, and last sequence number heard from the source through the upstream node.

- **Membership table:** The multicast group information is stored in the Membership Table that is created and maintained by each node for the source S, and multicast *groupId*. The Membership Table contains the node's status for a particular *tree t*. The status of a node can be any of pure forwarder, destination, and both forwarder and destination.

- **Message cache:** The message cache is generated and maintained by each node to detect duplicated packets.

- **Timeslot table:** Each node in the network creates and maintains a Timeslot table that contains the status of timeslots (*free*, *reserved*, *candidate*). The status of timeslots is exchanged between one-hop neighbor nodes using *Hello* message.

4.4 Centralized Multiple Disjoint Multicast Trees Routing Protocol (Centralized MDMTR)

Centralized MDMTR is an on-demand video multicast routing protocol that constructs multiple multicast trees and assigns MD video to the multicast trees in a centralized way. The construction of multiple multicast trees and the assignment of MD video are performed by three-way handshaking approach (Route Request (*RouteReq*), Route Reply (*RouteRep*), and Tree Construction (*TreeConst*) messages).

4.4.1 Route Discovery

When a multicast source node receives a request from the application layer to set up a QoS multicast connection to a group of destination nodes with bandwidth requirements for each *QoS_level*, it initiates a *RouteReq* message and floods it to its neighbors, as seen in Figure 4.16(a). The *RouteReq* message contains the following fields: (*source*, *request_id*, *type*, *route*, *free_timeslot*, *Bw_reqs*, *TTL*, *hop_count*), where (*source*, *request_id*) is used to uniquely identify a message. The *request_id* is monotonically increasing, which can be used to detect stale cash route. The type refers to message type. The *route* records the path from source to current traversed node. The *free_timeslot* records the status of slot assignment on the route. The *hop_count* is initially set to zero. The *TTL* is used to limit the hop count of the path.

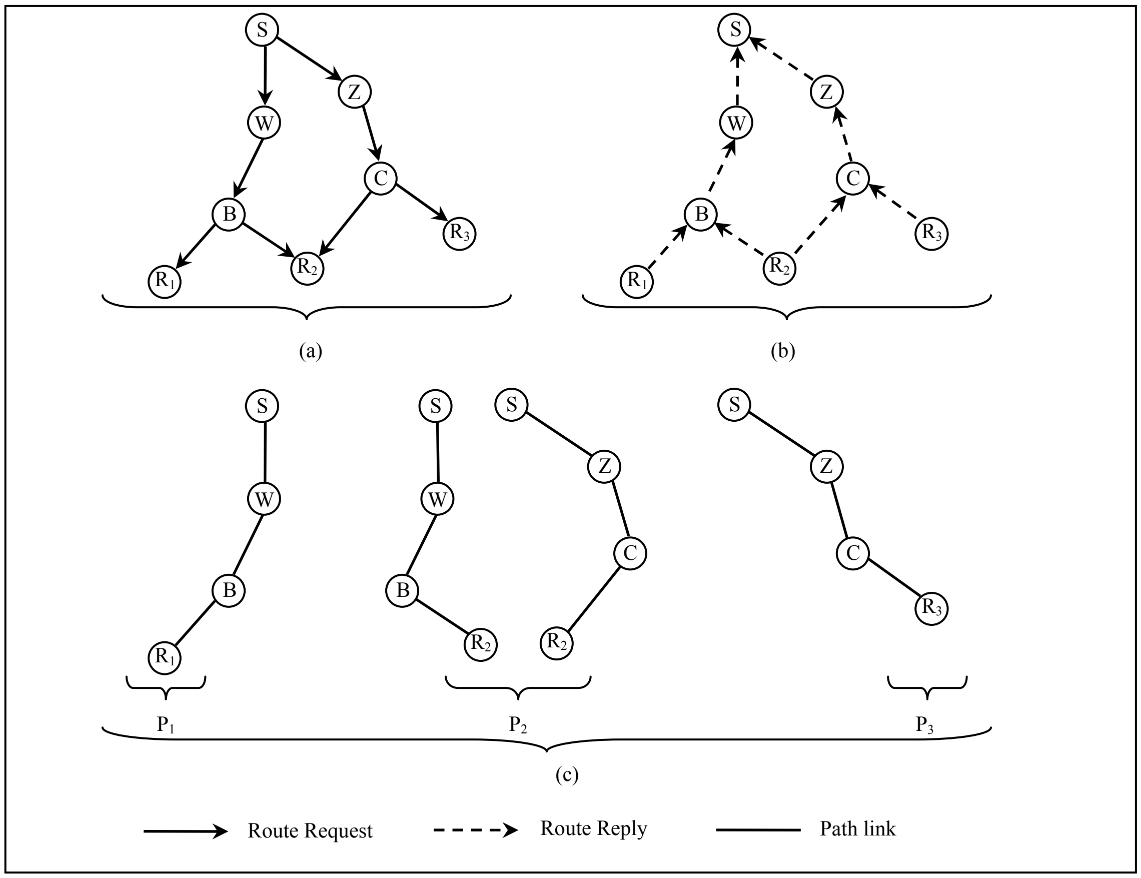


Figure 4.16 Route discovery in Centralized MDMTR protocol.

When a forwarding node, i.e., nodes $W, Z, B,$ and C in Figure 4.16(a), receives a non-duplicate *RouteReq* message, it checks if there are any common free timeslots between itself and the last node that sends the *RouteReq* message. If not, it means that there is no bandwidth to receive from the last node that sends the *RouteReq* message. Therefore, the *RouteReq* message is dropped. Otherwise, it appends its address, and its free timeslots information to the *RouteReq* message and it then re-broadcasts the message. This operation is repeated node by node until the value of TTL is reduced to zero. In order to increase the number of disjoint paths, a forwarding node will re-broadcast a duplicate *RouteReq* message that traversed through a different incoming link than the link from which the first *RouteReq* message is received, and whose hop count is not larger than that of the first received *RouteReq* message.

4.4.2 Route Selection and QoS_level Determination Phases

When a destination node receives a *RouteReq* message, it checks if there are any common free timeslots between itself and the last node that sends the *RouteReq* message. If not, it drops the *RouteReq* message. Otherwise, it records this path. If the destination node has a *QoS_level one*, i.e., it has no more free timeslots, it directly unicasts a *RouteRep* message to the multicast source S on the reverse path. Each node on the reverse path receives this *RouteRep* message, it marks its timeslots recorded in the *RouteRep* message as *candidate*. This process continues until the *RouteRep* message reaches the multicast source S. The timeslot status at each node will remain in *candidate* status until the node receives a *TreeConst* message from the multicast source S. If no *TreeConst* message arrives at the node, the route entry will be deleted and the status of timeslot will be marked as *free*.

When the destination node has still more free timeslots and it needs more video descriptions, it will not directly unicasts the *RouteRep* message to the multicast source S. It will then wait either for a short time or the reception of a certain number of *RouteReq* messages. When the destination node receives a proper number of *RouteReq* messages or after a timeout, it will sort all disjoint paths in descending order according to their number of hops and then selects the proper paths. After that it sends a *RouteRep* message to the multicast source S for each selected paths (refer to Figure 4.16 (b)). The *RouteRep* message is treated as mentioned before.

Each destination can determine its *QoS_level* based on its number of disjoint paths discovered during the route discovery phase. Note that if a destination node has three free timeslots, but only two disjoint paths are discovered then its *QoS_level* is equal to *two*. This means that the *QoS_level* does not depend only on the available bandwidth (i.e., the number of free timeslots) at a destination node, but also it depends on the number of disjoint paths discovered.

Figure 4.17 shows that a destination node R has two disjoint paths discovered during the route discovery phase. Its free timeslots are $\{ts_1, ts_2, ts_3\}$. The timeslots ts_2 and ts_3 will be assigned to the links (B, R) and (A, R), respectively. As a result, the destination R still has one free timeslots (ts_1) but it cannot require *QoS_level three* because it has only two disjoint paths. Therefore its *QoS_level* is equal to two. The *QoS_level* for any destination is determined by:

$$QoS\_Level = N(P_{R_i}) \qquad (11)$$

where $N(P_{R_i})$ is the number of discovered disjoint paths to a destination R_i.

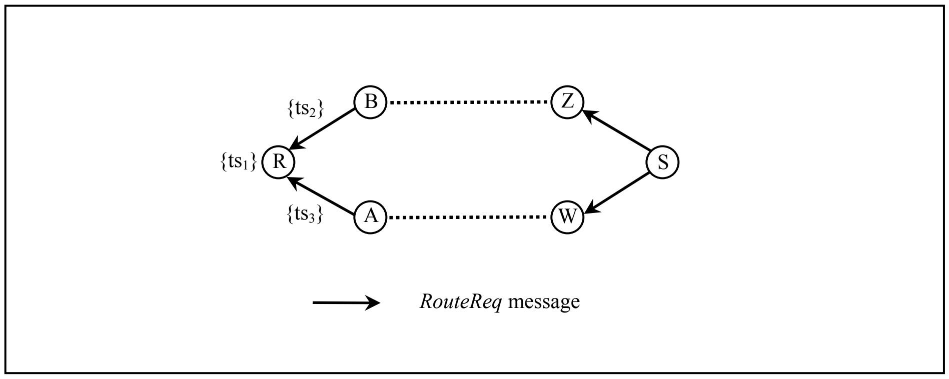


Figure 4.17 QoS_Level determination.

4.4.3 Multicast Trees Construction and Video Descriptions Assignment

After a pre-specified timeout, if the multicast source node cannot receive any more *RouteRep* messages from the destination nodes, the route discovery process completes. At this point, when the route discovery and reply phases are completed, the multicast source records the multiple disjoint paths for each destination R_i in set P_{R_i}, as seen in Figure 4.16(c). After that, it constructs multiple multicast trees according to algorithm 7 (see chapter 3) and assigns different video descriptions to each multicast trees. Figure 4.18 shows the constructed multicast trees.

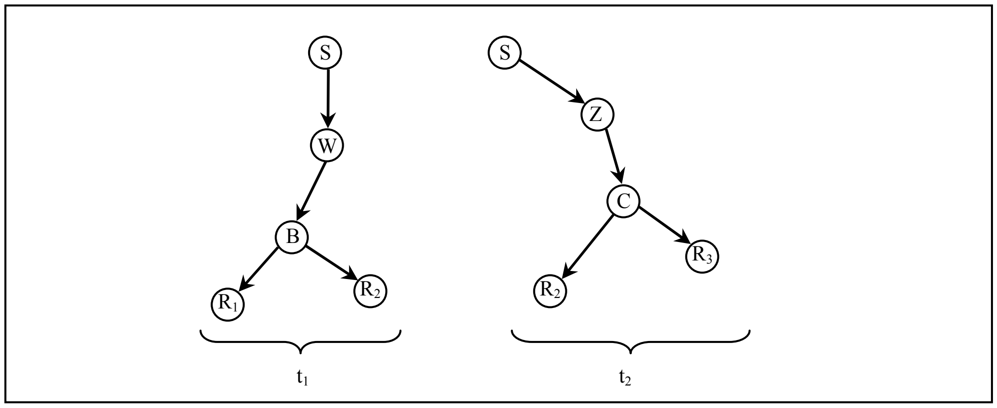


Figure 4.18 Multicast trees construction in Centralized MDMTR protocol.

When the source node completed multicast trees construction and video descriptions assignment, it sends this information using *TreeConst* messages to all destination nodes. When a node recieves a *TreeConst* message, it checks if its address is recorded in the *TreeConst* message. If not, the *TreeConst* is dropped. Otherwise, it marks its timeslots recorded in the *TreeConst* message as *reserved*, and records the source address, the multicast group address, and the multicast *tree t* in the routing table. It then re-broadcasts the *TreeConst* message. At the end of this operation, the multicast trees connection is established and the multicast source can begin transmitting video to destination nodes.

4.4.4 Multicast Trees Morphing

This phase can be divided into three phases: multiple multicast trees maintenance phase, leaving a multicast group phase, and joining a multicast group phase.

4.4.5 Multiple Multicast Trees Maintenance

As nodes in the network move or as wireless transmission conditions change, some nodes (e.g., forwarders or destination members) may become disconnected from the multicast forwarding tree of the group. When a broken link is detected between two nodes on a multicast *tree t*, the two nodes should delete the link from their list of next hops for the multicast group and release all *reserved* timeslots for this link and mark them as *free*. The node which is further from the multicast source (i.e., the node downstream of the break) is responsible for initiating the repair of the broken link. The downstream node detects that it has become disconnected from the multicast *tree t* when it fails to receive a number of successive expected multicast video packet from its upstream node on the *reserved* data timeslot. The downstream node can recognize that it has not received a video packet during the *reserved* data timeslot based on an expected inter-arrival time for the application's packet. The expected packet inter-arrival time may be set to a default value, may be defined according to the port number indicated in the packet, or may be specified by the sending application if an Application Programming Interfaces (API) is available for this purpose (Jetcheva et Johnson, 2001).

Each forwarder or destination node for the multicast *groupId* and source S maintains a *Disconnection Timer*. The *Disconnection Timer* is refreshed each time a video packet is received. The *Disconnection Timer* is based on the expected packet inter-arrival time value of the last received packet, plus a delay proportional to the node's hop count from the multicast source S.

When a downstream node, K, detects a break (see Figure 4.19 (a)), it initiates a *local repair* for the multicast forwarding *tree* t_1. At first, node K sends a *Repair Notification* message to the other nodes on the sub-tree (nodes below node K) in the multicast distributed tree for source S, multicast *groupId*, and *tree* t_1. The *Repair Notification* message serves two

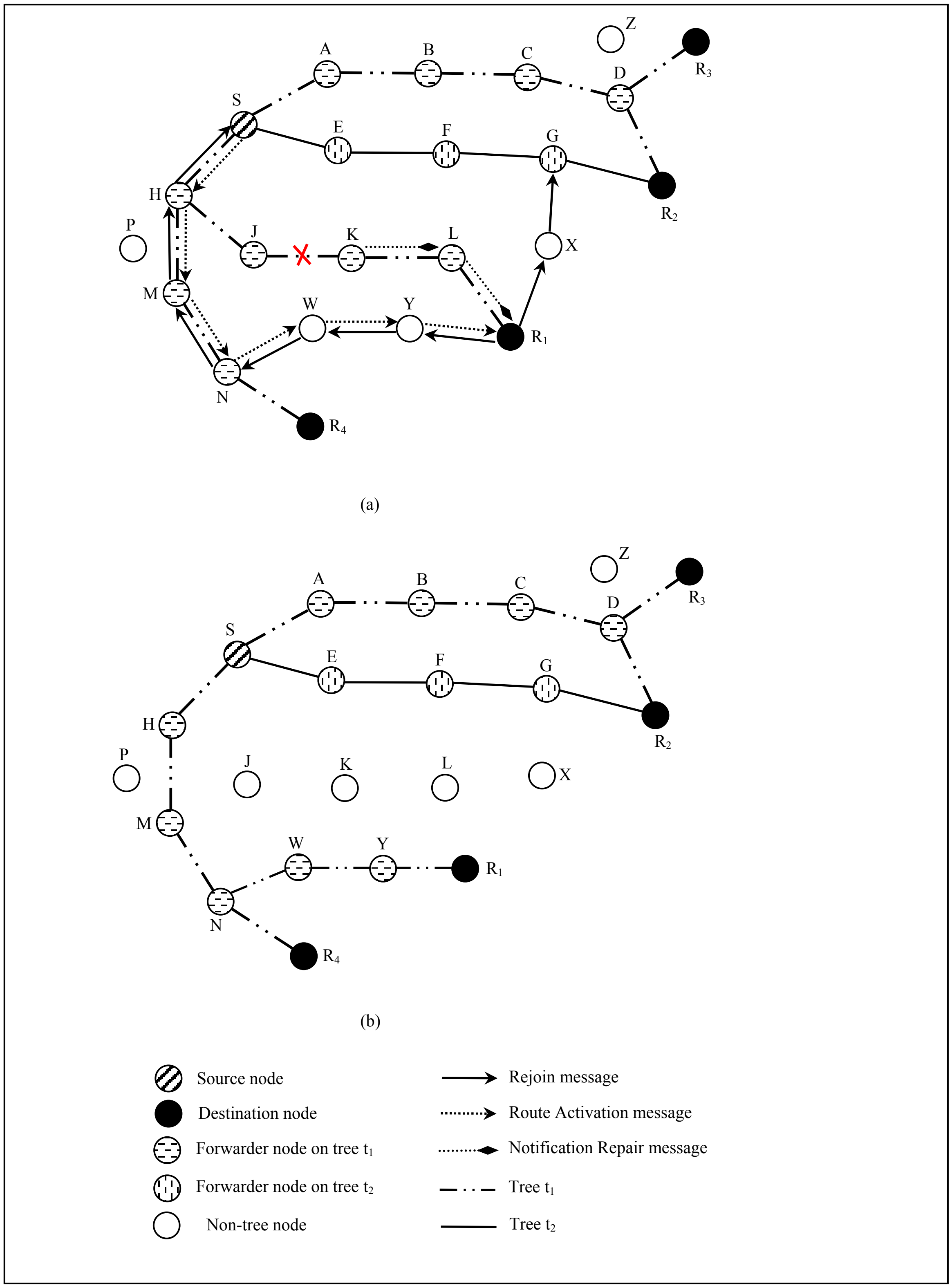


Figure 4.19 Multicast tree maintenance in Centralized MDMTR protocol.

purposes (Jetcheva et Johnson, 2001). It is a notification to nodes in the sub-tree below K that a *local repair* is in progress and that they should not initiate their own *local repair*. In addition, the *Repair Notification* message may be received by K parent's node (node J). If node J received the *Repair Notification*, it recognizes that one of its child nodes, node K, is performing a *local repair*. The node J then sends a *Repair Notification* message to node K, causing it to cancel its *local repair*.

When a destination node, node R_1 receives a *Repair Notification* message, or when, it initiates *local repair* by sending a *Repair Notification* message, it postpones its *Disconnection Timer* for a period of time (*Repair Delay*) equivalent to the *local repair* expected time.

After sending the *Repair Notification* message, node K waits for a short period of time (*Start Repair*) before it starts its *local repair*. If during *Start Repair* delay node K receives a *Repair Notification* message initiated by an upstream node for the same *tree* t_1, multicast *groupId*, and source S, then K cancels its *local repair*, since the repair should be performed by the downstream node that is adjacent to the broken link.

After *Start Repair* time has expired, and node K has not received a *Repair Notification* message initiated by an upstream node, for *tree* t_1, source S, and multicast *groupId*, it initiates a TTL-limited (e.g., TTL = 2) *Route Repair* message with source S, multicast *groupId*, *tree* t_1, and the *hop_count* from source S to node K. This *Route Repair* message is broadcasted as a form of network flooded.

A node receiving this *Route Repair* message can respond if it is a member of the multicast *tree* t_1, its hop count to the multicast source is less than or equal to that contained in the *Route Repair* message, and it has common free timeslots between itself and the last node that sends the *Route Repair* message. If the originating node receives more than one *Repair Ack* messages, the node selects the *Repair Ack* message with minimum hop counts to the multicast source and unicasts a *Route Activation* message to the selected route to activate it.

Since the node was repairing a tree break, it is likely that it is now a different distance from the multicast source than it was before the break. If this is the case, it must inform its sub-tree below of their new distance from the multicast source.

If the *local repair* procedure described above succeeds, the multicast forwarding sub-tree will be reestablished and the destination node will continue to receive multicast video packet as expected. Otherwise, the destination node will rejoin the multicast *tree* t_1 as follows. When the *Disconnection Time* expires at a destination node R_1, it means that the *local repair* has probably failed. In this case, the destination node R_1 in Figure 4.19(a) should rejoin the multicast *tree* t_1. Node R_1 initiates and broadcasts a *Rejoin* message to the multicast source S, multicast *groupId*, and multicast *tree* t_1. Non-member nodes (a node that is not a member of multicast *groupId*, source S, and does not belong to *tree* t_1 or *tree* t_2), nodes X and Y, can rebroadcast the *Rejoin* message if they have common free timeslots with the last node that sends the *Rejoin* message. Non-member nodes will continue to rebroadcast the *Rejoin* message until it reaches the multicast source S or a member node, node N, (a node that belongs to multicast source S, multicast *groupId*, and *tree* t_1).

When a member node, node N, receives a *Rejoin* message, it means that there is a path from the multicast source S to the destination that initiates the *Rejoin* message (node R_1). After that, node N, instead of broadcast the *Rejoin* message, unicasts the *Rejoin* message to its parent node, node M. Finally, the *Rejoin* message will reach the multicast source S. The source S may receive multiple *Rejoin* messages. In this case, it will select the one with shortest path and sends a *Route Activation* message to the destination node R_1. Eventually, the destination node R_1 will receive the *Route Activation* message and rejoin the multicast *tree* t_1. Figure 4.19(b) shows the multicast trees at the end of rejoining process.

4.4.6 Joining a Multicast Group

When a new destination node wishes to join a multicast group (node R_1 in Figure 4.20(a)), it initiates a Join Request (*JoinReq*) message with the destination address set to that of the multicast group, with its free timeslots and with hop count equal to zero and broadcasts it to its neighboring node. Any neighboring node (nodes Y, L, and X) receiving this *JoinReq* message will rebroadcast it if there are common free timeslots between itself and the node that sends this *JoinReq* message. This process will continue until the *JoinReq* message reaches the multicast source or a member node (forwarding or/and destination node on *tree* t_1 or *tree* t_2). When a member node receives this *JoinReq* message (nodes $G, H,$ and N), it checks if there is any common free timeslot between itself and the last node that sends the *JoinReq* message. If so, there is a path from the multicast source node to the node that initiated the *JoinReq* message, R_1. After that, a member node (nodes $G, H,$ and N) unicasts a *JoinReq* message to its upstream node. This *JoinReq* message will re-unicast until it reaches the multicast source S.

The multicast source may receive multiple *JoinReq* messages. It then selects the proper disjoint paths and unicasts *JoinRep* (Join Reply) messages on the reverse paths. The multicast source will select the shortest path for each multicast tree. For example, it will select the path $S \rightarrow H \rightarrow J \rightarrow K \rightarrow L \rightarrow R_1$, instead of the path $S \rightarrow H \rightarrow M \rightarrow N \rightarrow W \rightarrow Y \rightarrow R_1$, for the first tree ($t_1$) and the path $S \rightarrow E \rightarrow F \rightarrow G \rightarrow X \rightarrow R_1$ for the second tree (t_2). Therefore, the destination node R_1 will be assigned two video descriptions (its *QoS_level* is equal to *two*). Figure 4.20(b) shows the structure of multicast trees at the end of the joining process.

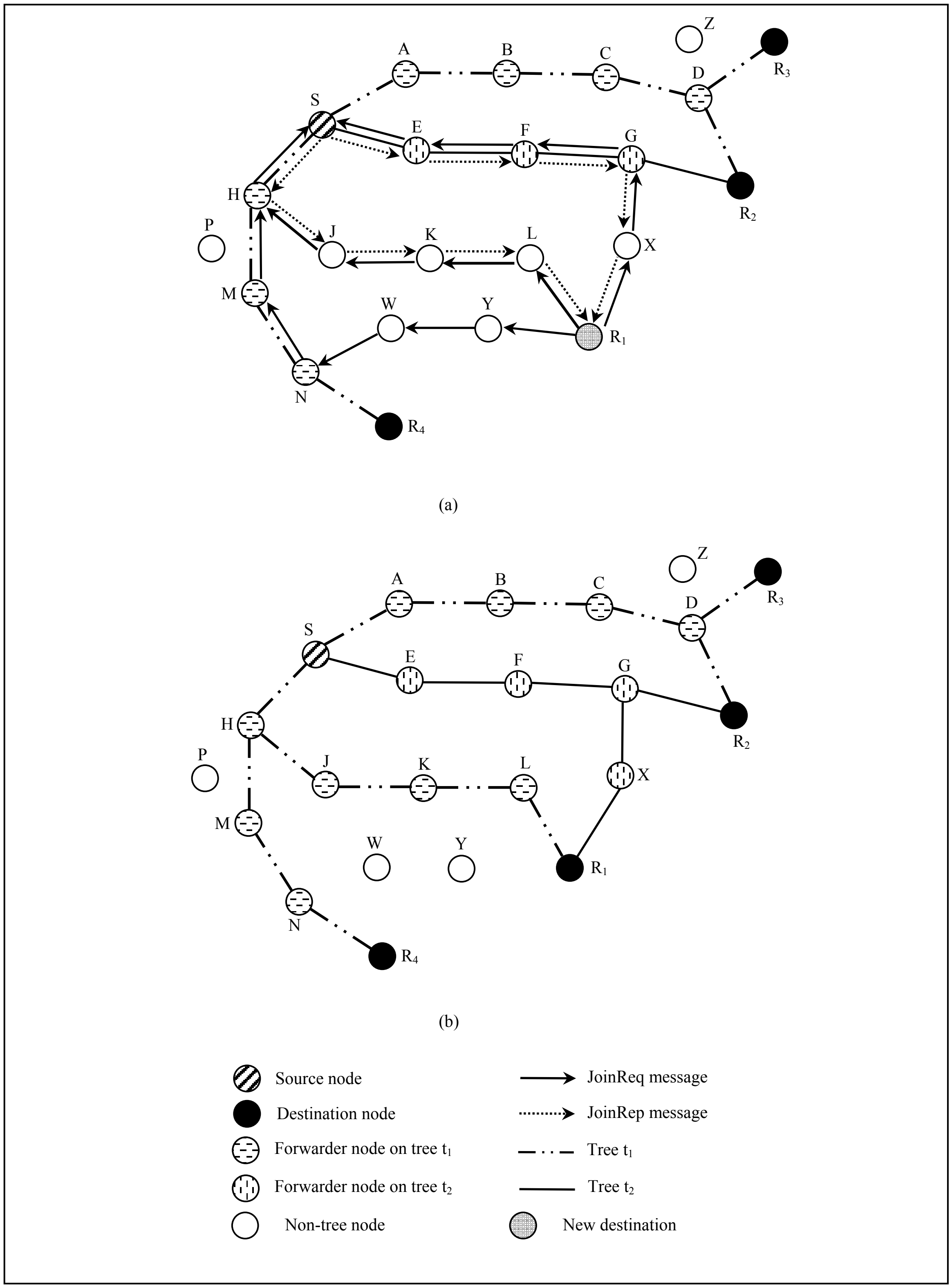


Figure 4.20 New destination joins a multicast group in Centralized MDMTR protocol.

4.4.7 Leaving a Multicast Group

When a leaf destination node wishes to leave the multicast group it initiates *Prune* messages and sends them to its upstream nodes and prune itself by deleting all information concerning the multicast group, i.e., source address, multicast group address. It then releases the *reserved* timeslots and marks them as *free*. If a destination is not a leaf node, it cannot leave the multicast group but it can mark itself as a forwarding node. When a node receives a *Prune* message, it checks in its routing table if it has a downstream node other than the node sending the *Prune* message. If it is the case, it cannot prune itself and therefore it stays connected to the tree and then drops the *Prune* message and releases its *reserved* timeslots (if they are not used for transmission to the other downstream nodes) for this link by marking them as *free*. Otherwise; it prunes itself and sends the *Prune* message to its upstream node. Furthermore, it will release its timeslots (for transmission and reception) for this session and will mark them as *free*.

This process continues until the existing *Prune* message arrives at the source node. If a source node receives a *Prune* message from its downstream node it deletes it from its routing table. After that the source checks if the common timeslots with the deleted downstream node are not *reserved* with its other downstream nodes. In that instance, it releases them and marks them as *free*. Otherwise; they stay *reserved*. The process of releasing the *reserved* timeslots gives the opportunity for other traffics to use them.

4.5 Sequential Multiple Disjoint Multicast Trees Routing Protocol (Sequential MDMTR)

Sequential MDMTR constructs multiple disjoint multicast trees and assigns MD video to the destination nodes in a centralized fashion. However, the main difference between sequential MDMTR and centralized MDMTR is that the assignment of MD video is executed in a sequential way. This means that all the destination nodes should be first assigned the first description, then the destination nodes that have *QoS_level two* should be assigned the second description and the destination nodes that have *QoS_level* three should be assigned the third description and so on. Therefore, to perform the assignment of MD video in a sequential way, the destination nodes on each multicast tree should be superset of the later, i.e., $t_L \subseteq t_{L-1} \ldots \subseteq t_2 \subseteq t_1$. Algorithm 7 (in chapter 3) is deployed to construct multiple disjoint multicast trees, and then algorithm 9 is executed to form the final version of the multiple multicast trees. After that, the trees $t_1, t_2, \ldots, t_L$ will be assigned the first, the second and the L-*th* description, respectively. It is worth mentioning that Serial MDTMR assigns MD video to the destination nodes in a sequential way but in a distributed manner, as we explained previously.

We use Figure 4.16, to explain how sequential MDMTR constructs multiple disjoint multicast trees. At the end of algorithm 7, two disjoint multicast trees are constructed, namely, t_1 and t_2 as seen in Figure 4.18. However, in order to perform sequential assignment of MD video, R_3 should be connected to t_1. And because Sequential MDMTR maintains totally disjoint multicast trees, therefore, only one multicast tree, t_1, is constructed as shown in Figure 4.21.

```
1. for $i$ = 1 to $L$
2.    Add the destination nodes to each $t_i$ such that:
          $t_L \subseteq t_{L-1} \subseteq \ldots \subseteq t_2 \subseteq t_1$
3. end for
```

Algorithm 9 Sequential MDMTR: Multicast Trees Construction.

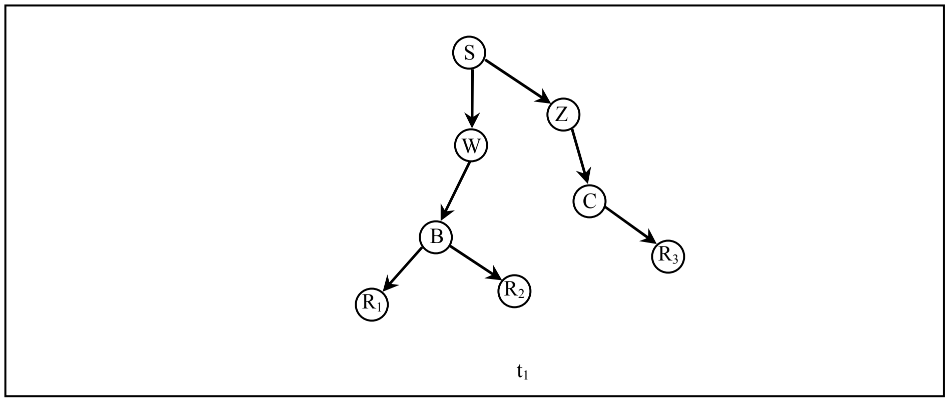


Figure 4.21 Sequential MDMTR: Multicast tree construction.

4.6 Distributed Multiple Disjoint Multicast Trees Routing Protocol (Distributed MDMTR)

Distributed MDMTR assigns MD video to the nodes and constructs multiple disjoint multicast trees in a distributed way. The construction of multiple multicast trees and the assignment of MD video are performed by two-way handshaking approach (Route Request (*RouteReq*) and Route Reply (*RouteRep*) messages). Compared with centralized MDMTR, distributed MDMTR offers minimum construction delay and routing overhead.

The main difference between Centralized MDMTR and Distributed MDMTR is the assignment of MD video and the construction of multiple disjoint multicast trees. However, in Distributed MDMTR, multicast tree maintenance, leaving a multicast group, and joining a multicast group are performed in the same way as in Centralized MDMTR.

4.6.1 Multicast Trees Construction and MD Video Assignment

As in Centralized MDMTR, when a multicast source node, in distributed MDMTR, receives a request from the application layer to set up a multicast connection, it broadcasts a *RouteReq* message to its neighbors. The multicast source appends its address, multicast *groupId*, its *free* timeslots, MD video available and the bandwidth requirements for each description in terms of number of timeslots. When a neighbor node receives the *RouteReq* message it checks if there are any common free timeslots between itself and the multicast source, if yes it indicates that this node can be a member of the multicast forwarding group. It then randomly selects one description and rebroadcasts the *RouteReq* message to its neighbors after it appends (in the *RouteReq* message) its address, *free* timeslots, and the video description that has been selected.

When one of its neighbor nodes receives this *RouteReq* message, it checks if there are any common *free* timeslots between itself and the last node that sends the *RouteReq* message. In the affirmative it rebroadcasts the *RouteReq* message after it appends its address, and its *free* timeslots. Each node that has forwarded the *RouteReq* message should record in its routing table the multicast source address, the multicast *groupId*, and the video description that is recorded in the *RouteReq* message. In order to maintain totally disjoint multicast trees, each node should rebroadcast only one *RouteReq* message, therefore when a duplicate *RouteReq* message is received the node will drop it. This process will continue until the *RouteReq* message reaches a destination node. When a destination node receives a *RouteReq* message it checks if there are any common *free* timeslots between itself and the last node that sends the *RouteReq* message, if yes, it records in its routing table the information recorded in the *RouteReq* message. If the destination still has more *free* timeslots this means that it can request more descriptions, therefore it will wait for a short time to select a proper path for each description. After a timeout, the destination unicasts a *RouteRep* message to the multicast source for each selected path.

After a timeout, the multicast source will receive multiple *RouteRep* messages from destination nodes. It will then construct multiple disjoint multicast trees as follows. All nodes that have selected the same description will be on the same tree. Therefore, multiple disjoint multicast trees are constructed. When the multicast source constructs multiple disjoint trees, it starts video transmission to the destination nodes. Figure 4.22 depicts the assignment of MD video and the construction of multicast tree.

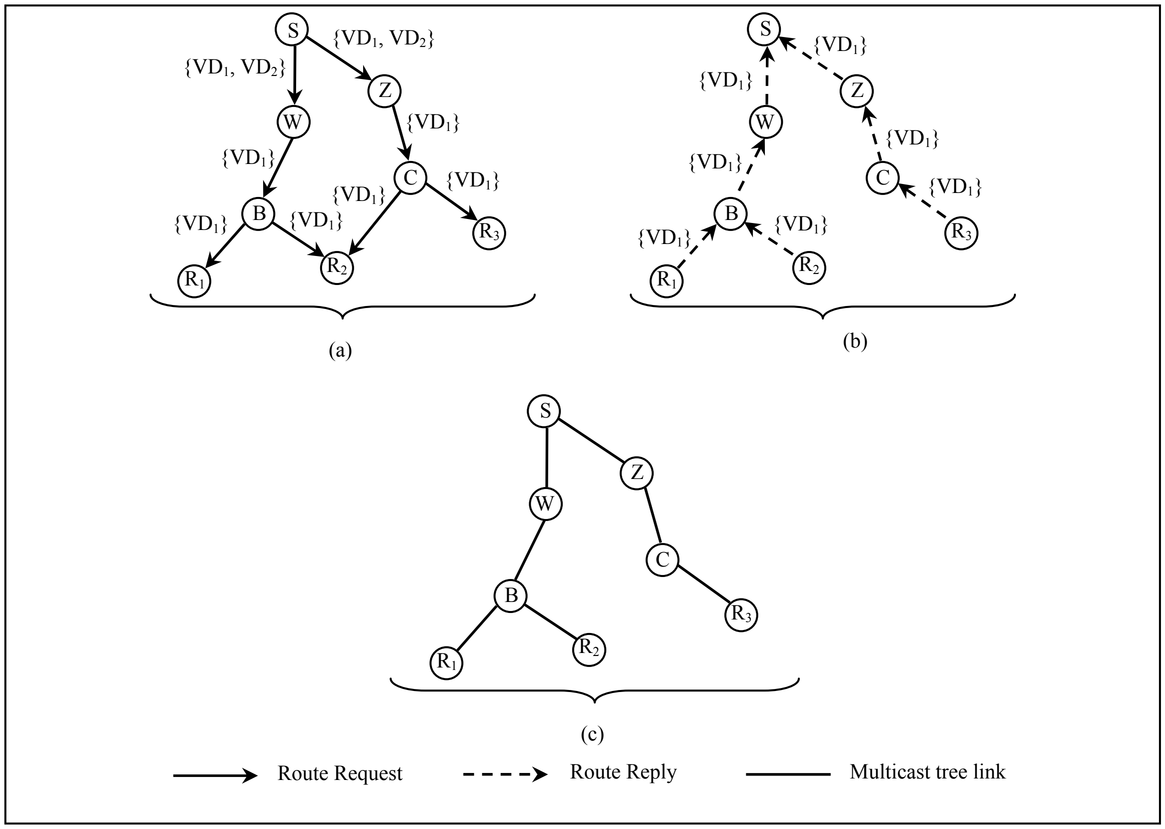


Figure 4.22 Multicast trees constructions and MD video assignment in Distributed MDMTR. (a) Broadcasting RouteReq message. (b) Uincasting ReouteRep message. (c) Multicast tree construction.

4.7 Neighbor-aware Multiple Disjoint Multicast Trees Routing Protocol

Neighbor-aware MDMTR is a variant of Distributed MDMTR protocol. Neighbor-aware MDMTR protocol assigns MD video and constructs multiple node-disjoint trees in a distributed manner. In Neighbor MDMTR each node waits for a short time (*JoinReq_Agg*) before it rebroadcasts a *JoinReq* message. This short time gives the nodes to decide which video description they should select to propagate. However, in Distributed MDMTR nodes select a random video description to rebroadcast. Next section presents how multiple disjoint multicast trees are constructed and how MD video are assigned.

4.7.1 Multicast Trees Construction and MD Video Assignment

When a multicast source node receives a request from the application layer to set up a multicast connection, it broadcasts a *RouteReq* message to its neighbors. The multicast source appends its address, multicast *groupId*, its *free* timeslots, MD video available and the bandwidth requirements for each description in terms of number of timeslots. When a neighbor node receives the *RouteReq* message it checks if there are any common free timeslots between itself and the multicast source, if yes it indicates that this node can be a member of the multicast forwarding group. It then waits for a short time (*JoinReq_Agg*) to receive multiple *RouteReq* messages from neighbor nodes. This time enables the nodes to decide which video descriptions they should select. For example, if a node, say X, hears its neighbors select the first video description it then selects the second video description to rebroadcasts. Figure 4.23 illustrates how Neighbor-aware MDMTR protocol assigns the video descriptions and constructs multiple multicast trees.

The source node S broadcasts a *RouteReq* message to its neighbors nodes, nodes W and Z, as seen in Figure 4.23(a). The *RouteReq* message contains the video descriptions available at the source node. There are two video descriptions VD_1 and VD_2. Assume node W selects VD_1. After that, it indicates the video description, VD_1 in the *RouteReq* message and rebroadcasts it to its neighbor nodes, nodes B and Z.

When node Z receives the *RouteReq* message from node W, it then selects the second video description, VD_2, in stead of random selection as in Distributed MDMTR. This process continues until the *RouteReq* message reaches the destination nodes. After that, each destination node select the appropriate paths and unicasts a *RouteRep* message to each selected path as shown in Figure 4.23(b).

After a timeout, the multicast source will receive multiple *RouteRep* messages from destination nodes. It will then construct multiple disjoint multicast trees as follows. All nodes that have selected the same description will be on the same tree. Therefore, multiple disjoint multicast trees are constructed. When the multicast source constructs multiple disjoint trees (see Figure 4.23(c)), it starts video transmission to the destination nodes.

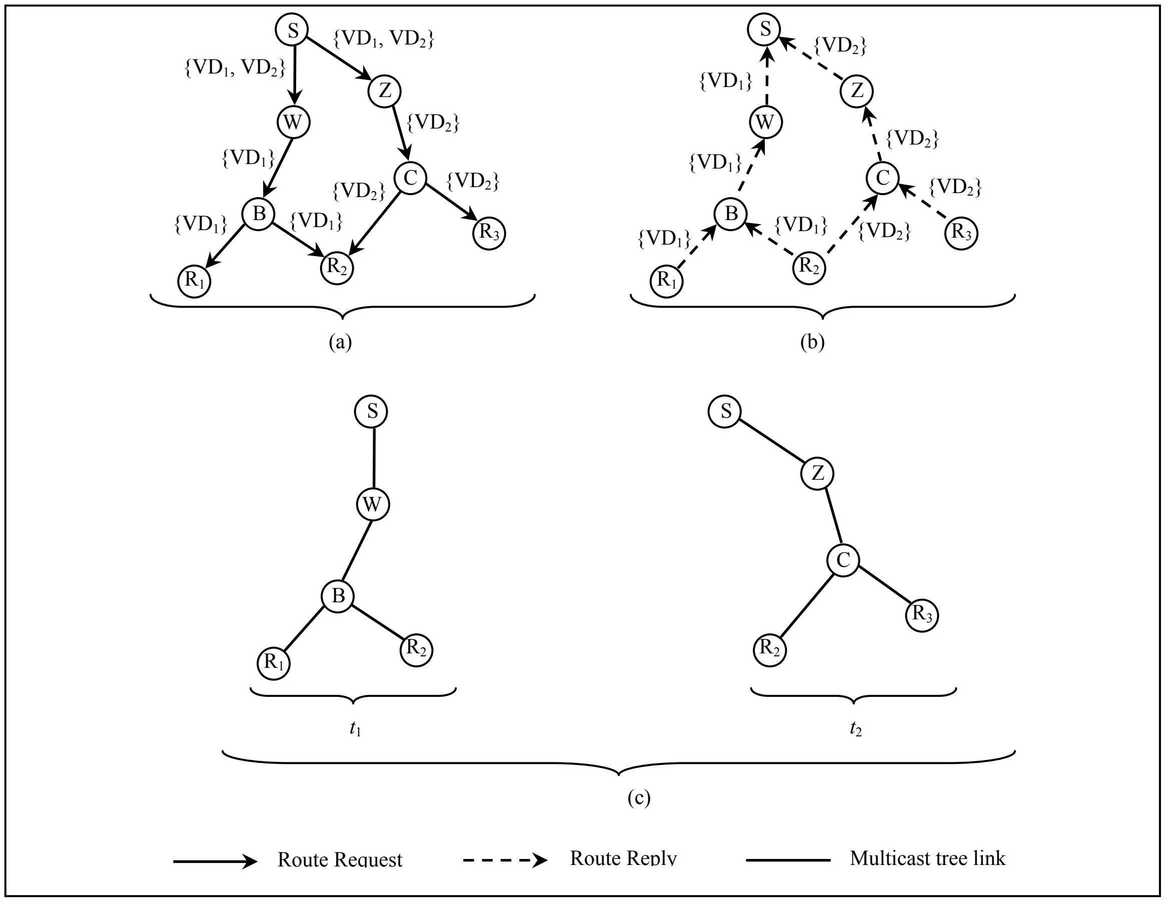


Figure 4.23 Video descriptions assignment and multicast trees construction in Neighbor-aware MDMTR.

4.8 Simulation Results

4.8.1 Simulation Framework

We compare the performance of our proposed protocols: Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR with that of Serial MDTMR (Wei et Zakhor, 2007). Serial MDTMR, based on ODMRP (Lee, Su et Gerla, 2002), constructs two node-disjoint multicast trees in a serial manner as described in chapter 2. The trees are numbered as *tree* t_1 and *tree* t_2. Using MDC, Serial MDTMR codes each video frame into two descriptions. Each description is transmitted along one tree.

However, serial MDTMR does not provide QoS capability. Furthermore, it does not take into consideration the heterogeneity of destination nodes. To make a fair comparison, we offer the QoS-extension Serial MDTMR such that each path in the Serial MDTMR protocol adopts Lin's QoS unicast path routing (Lin et Liu, 1999; Lin, 2001), where MAC sub-layer adopts CDMA over TDMA channel model. Furthermore, to enable Serial MDTMR to consider heterogeneous destinations, only destinations that have enough bandwidth will respond to the *Join Request* messages of the second tree (*tree* t_2). As a result, all destinations will be connected to *tree* t_1 while *tree* t_2 will have only the destinations that are capable of receiving the second description. Figure 4.24 shows how Serial MDTMR constructs multiple multicast trees for heterogeneous destinations.

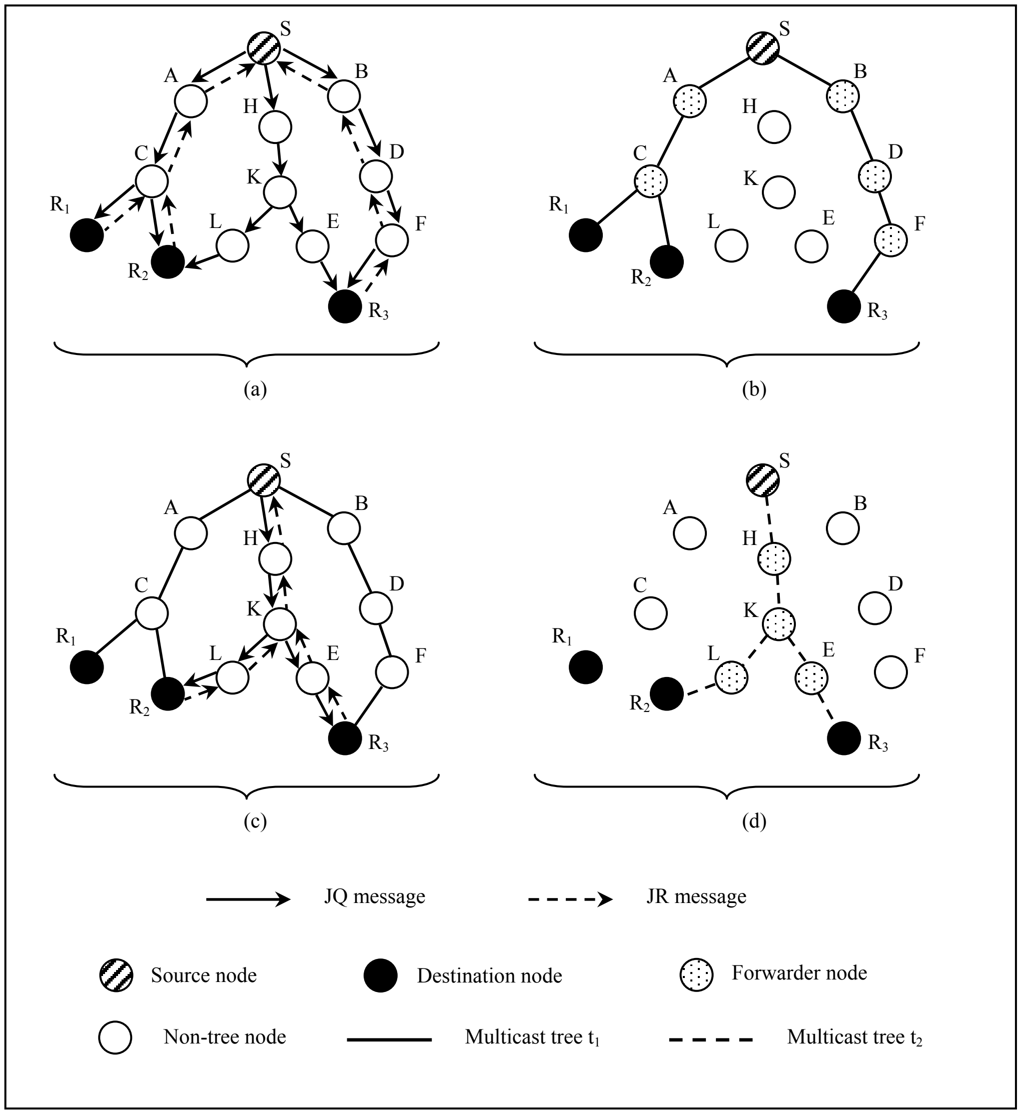


Figure 4.24 Multicast trees construction for heterogeneous destinations in Serial MDTMR: (a) First round of JQ message. (b) First multicast tree construction. (c) Second JQ message. (d) Second multicast trees construction.

4.8.2 Simulation Scenario

We have developed a simulator with MATLAB. The transmission rate is 2 Mbps, and the transmission range is 250 *m*. In each frame, the data slot in the data phase is set to 5 *m*s. The total number of slots in the data phase is set to 16. The control slot in control phase is set to 0.1 *ms* and the total number of slots in the control phase is set to 50. Each node in the network is equipped with a match filter receiver to detect the transmitted bits. The random waypoint model is used to model the mobility of the nodes (Broch et al., 1998). Each node is randomly assigned with an initial location, a destination and a speed. The speed is uniformly distributed between 0 and *maximum* speed. During the simulation, each node starts its journey from its initial location to the destination at the assigned speed. Upon reaching the destination, another random destination is targeted after a *pause time*. We only consider the continuous mobility case with zero *pause time*. The parameters setting for Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR are as follows: *packet inter-arrival time* is set to 100 *ms*, *start local repair* is set to 100 *ms*, *local repair* TTL is set to 2, *missing packet to trigger disconnection* is set to 2, *hello message interval* is set to 1 *s*, and *repair delay* is set to 100 *ms*. Serial MDTMR the parameters setting are as follows: *JoinRequest interval* is set to 3 *s*, and *forwarding state lifetime* is set to 4.5 *s*.

Our simulation setup consists of 50 nodes randomly spread in a rectangular terrain of area 1500 x 300 m^2. To change the mobility level of the network, we vary the *maximum* speed from 3 *m/s* to 18 *m/s*. Each simulation runs for a period of 900 *s*. The results are averaged over 30 simulation runs. The scenarios are generated prior to the simulation so that identical scenarios can be reused for each case to ensure fairness in the simulation study. One video source and five destinations (each destination node is at least two-hop away from the source) are randomly selected among 50 nodes and recorded so that the same nodes are used for each case to maintain fairness of the comparison study. The raw video is encoded into two descriptions. Each video frame is encoded into two packets using matching pursuits multiple description coding (MP-MDVC) (Tang et Zakhor, 2002) at 65 kbps. The frame rate is set to be 8 fps. The EvalVid toolset was used to obtain the encoded video traces in order to determine the size of the video packets to feed in the simulation (Klaue, Rathke et Wolisz, 2003). We consider interactive video applications in which the playback deadline of each packet is 150 *ms* after it is generated. If a packet is not received within its playback deadline it is considered lost.

In Centralized MDMTR, the number of video descriptions required by a destination node, $\mathbf{N}_{req}(R_i)$, is determined as we explained previously in Section 4.4.2. We use the same value of $\mathbf{N}_{req}(R_i)$ for Sequential MDMTR, Distributed MDMTR, Neighbor-aware MDMTR and Serial MDTMR protocols. The bandwidth requirement for each description is set to one timeslot.

4.8.3 Performance Metrics

We evaluated the performance of Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR and compared it to that of Serial MDTMR using the following metrics:

- **The ratio of user satisfaction (RUS):** refer to Equation 8.

- **Number of pure forwarder (PF):** It is defined as the number of pure forwarders on the multiple multicast trees that are not destinations. This measures the efficiency in terms of minimizing the number of pure forwarding nodes, as seen in Equation 9.

- **The ratio of bad frames (RBF):** It is defined as the ratio of the number of bad frames experienced in all the destinations to the total number of frames that should have been decoded in all the destinations.

$$RBF = \frac{\sum_{i=1}^{m} \mathbf{N}_{bf}(R_i)}{\sum_{i=1}^{m} \mathbf{N}_{f}(R_i)} \qquad (12)$$

where, $\mathbf{N}_{bf}(R_i)$ and $\mathbf{N}_{f}(R_i)$ represent the number of bad frames and the number of frames that should have been decoded of a destination R_i, respectively.

- **The number of bad periods:** A bad period consists of contiguous bad frames. This metric reflects the number of times that received video is interrupted by the bad frames.

- **Normalized packet overhead:** It is defined as the total number of data and control packets generated by the network divided by the total number of data packets actually received. This measures both the data forwarding efficiency and also the control overhead of the multicasting protocol.

- **Control overhead:** It is defined as the total number of control packets generated by the network divided by the total number of successfully decoded video frames at each destination.

Varying Number of Multicast Destinations

Figure 4.25, Figure 4.26, Figure 4.27, Figure 4.28, Figure 4.29, and Figure 4.30 illustrate Centralized MDMTR's, Sequential MDMTR's, Distributed MDMTR's, Neighbor-aware MDMTR's, and Serial MDTMR's performance with varying number of multicast destinations. We use the simulation setup described in Section 4.8.2 with node mobility 3 *m/s*. The number of destinations was varied from 5 to 26.

In Figure 4.25, one can see that Centralized MDMTR protocol achieve a higher user satisfaction compared to the other protocols. Furthermore, it achieves a 100% of user satisfaction when each destination nodes requests a number of MD video equals to its number of disjoint paths. As the number of destinations increases, the user satisfaction does not change (good scalable). However, when the *independent-description* property is taken into consideration (e.g., Sequential MDMTR and Serial MDTMR protocols), the user satisfaction degrades gradually as the number of destination nodes increases. Sequential MDMTR protocol achieves a higher user satisfaction compared to Serial MDTMR protocol. Sequential MDMTR protocol constructs multiple multicast trees and assigns MD video in a centralized manner. On the other hand, Serial MDTMR protocol performs that in a distributed manner. However, when the multicast trees construction and the assignment of MD video performed in a distributed manner this may blocked some paths of destination nodes to be assigned different video descriptions. It is more efficient if the multicast trees construction and the assignment of MD video are performed in a centralized manner, e.g. Sequential MDMTR protocol. In sequential MDMTR, and Serial MDTMR the set of destination nodes on the previous tree is the superset of the later (i.e., *tree* $t_2 \subseteq$ *tree* t_1). This means that all the destination nodes of *QoS_level one* should be assigned the first description and should be connected to the same tree.

Distributed MDMTR and Neighbor-aware MDMTR protocols assign MD video and construct multiple multicast trees in a distributed manner. In spite they reduce the delay of multicast trees construction, the number of pure forwarders nodes (see Figure 4.26), the control overhead (see Figure 4.29), and the normalized packet overhead (see Figure 4.30) but they achieve a lower user satisfaction compared to Centralized MDMTR and Sequential MDMTR protocols.

However, Distributed MDMTR and Neighbor-aware MDMTR protocols deploy the same concept (*independent-description* property of MDC) when they assign the video descriptions to the nodes, but they achieve a lower user satisfaction compared to Centralized MDMTR protocol. This is because the assignment of MD video is randomly performed. As a result, two paths for the same destination node may be assigned the same video description.

We can observe from that all the protocols almost achieve the same number of bad frames and the number of bad periods (refer to Figure 4.27 and Figure 4.28).

One can see that Serial MDTMR has a tremendous overhead since it uses a native flooding based tree construction (refer to Figure 4.29 and Figure 4.30). Distributed MDMTR has slightly lower overhead compared to the other protocols because it deploys two-handshaking for constructing and assigning MD video, and it has a minimum number of pure forwarders nodes.

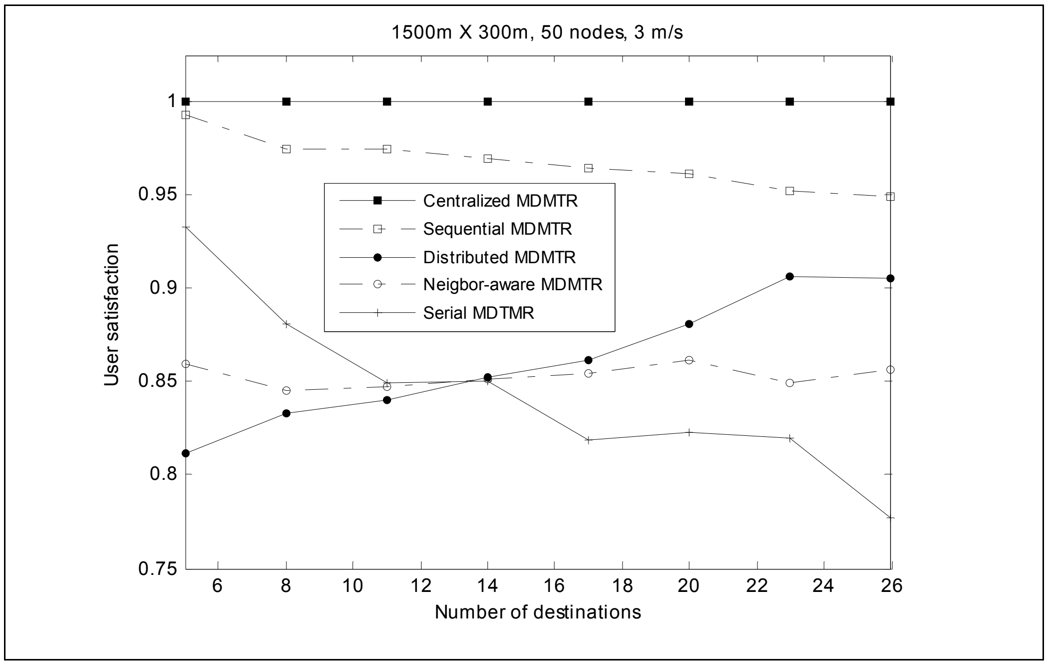


Figure 4.25 Performance evaluation for multiple multicast trees: User satisfaction vs. Number of destinations.

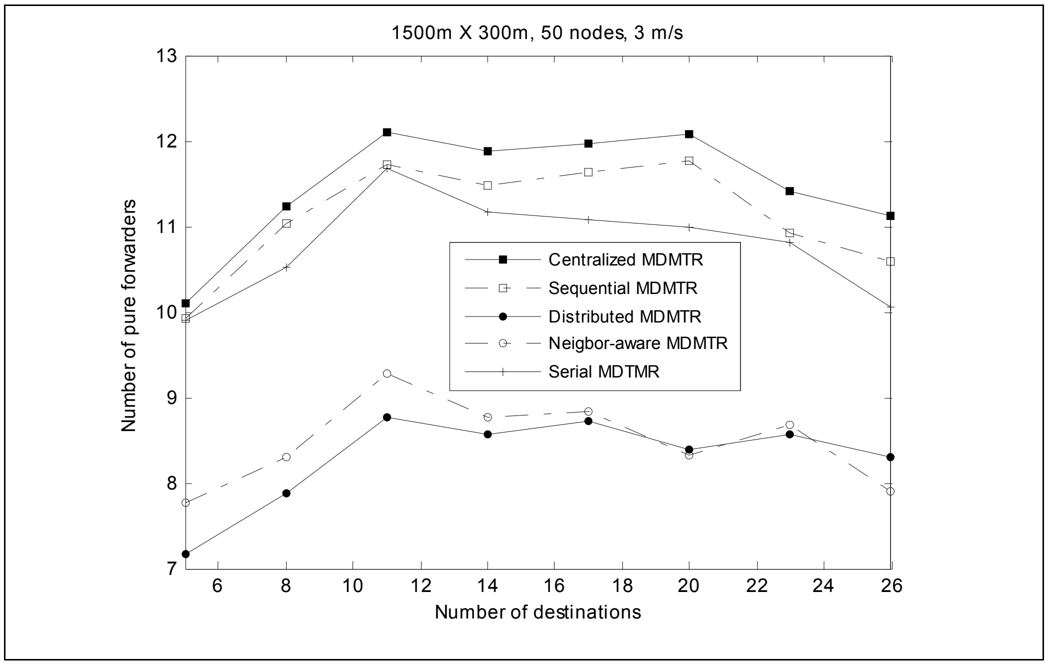


Figure 4.26 Performance evaluation for multiple multicast trees: Number of pure forwarder vs. Number of destinations.

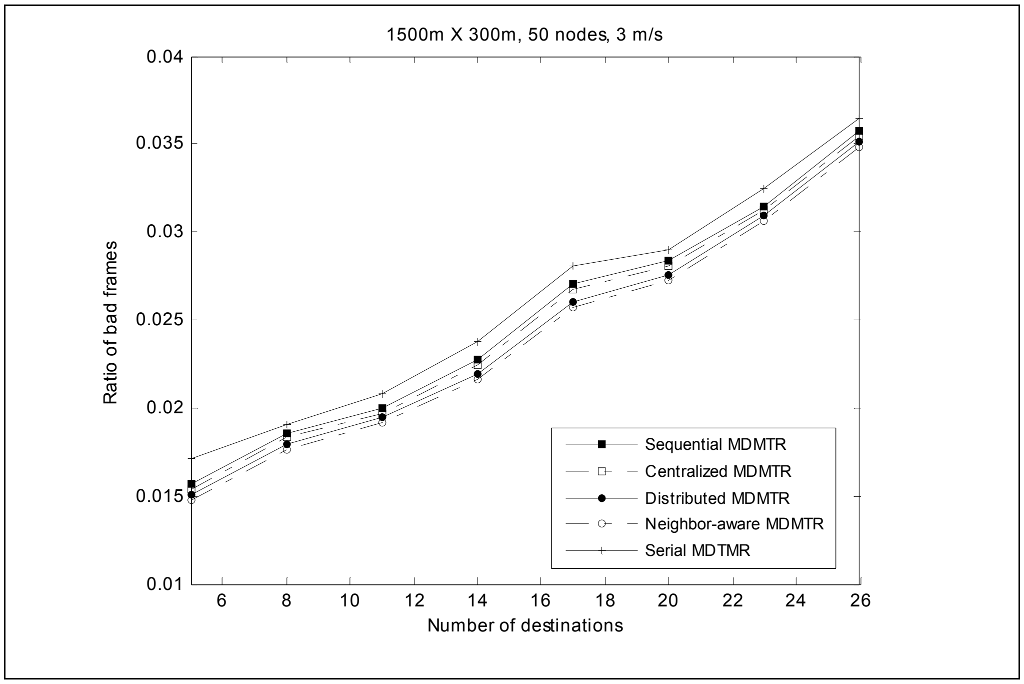


Figure 4.27 Performance evaluation for multiple multicast trees protocols: Ratio of bad frames vs. Number of destinations.

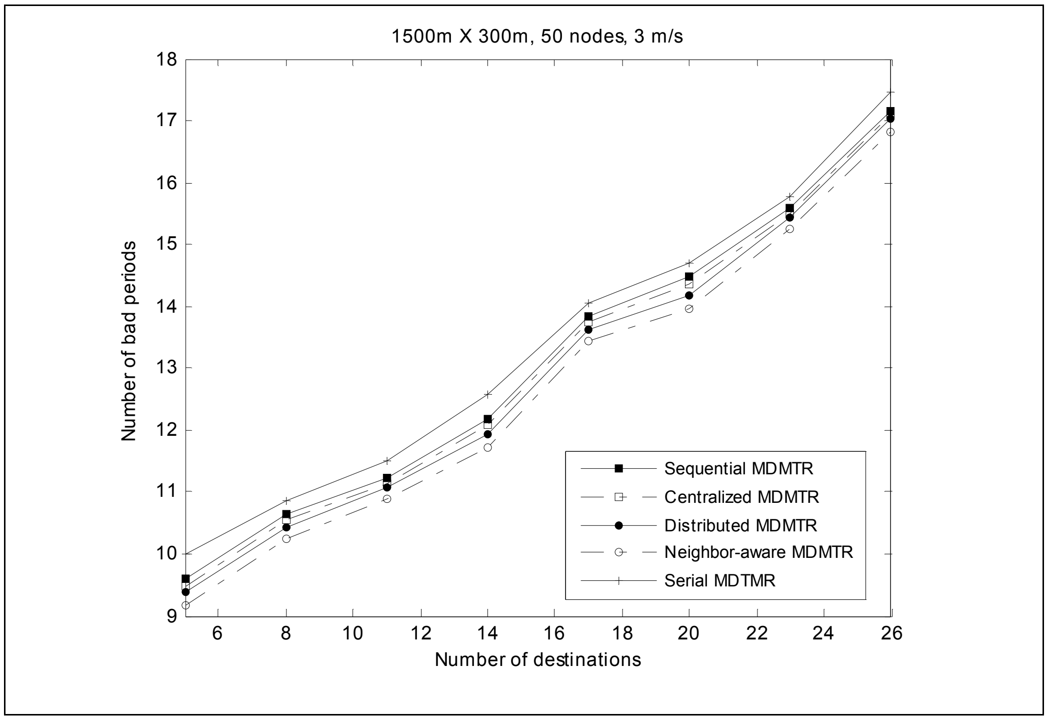


Figure 4.28 Performance evaluation for multiple multicast trees protocols: Number of bad periods vs. Number of destinations.

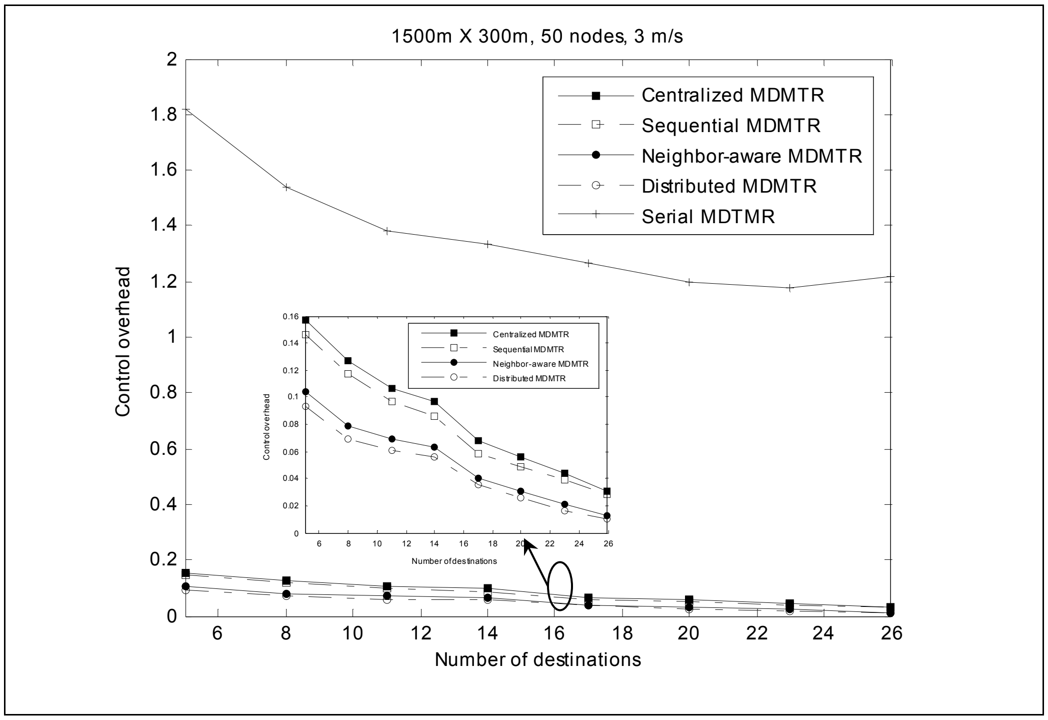


Figure 4.29 Performance evaluation for multiple multicast trees protocols: Control overhead vs. Number of destinations.

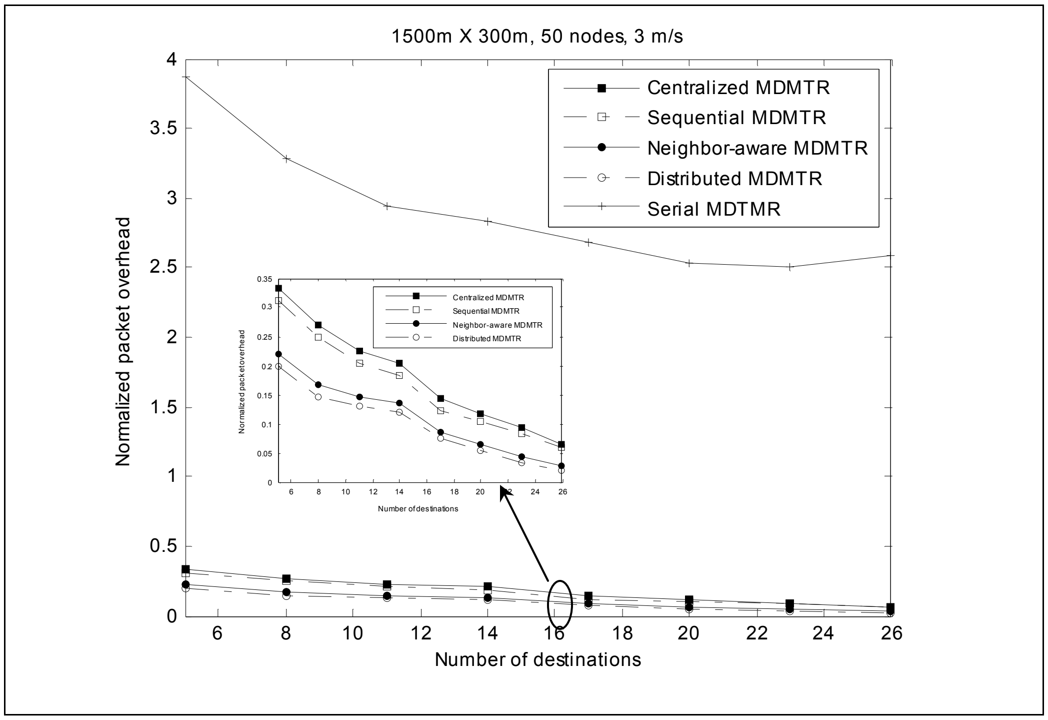


Figure 4.30 Performance evaluation for multiple multicast trees protocols: Normalized packet overhead vs. Number of destinations.

Varying Node Speed

Figure 4.31, Figure 4.32, Figure 4.33, and Figure 4.34 compare Centralized MDMTR's, Sequential MDMTR's, Distributed MDMTR's, Neighbor-aware MDMTR's, and serial MDTMR's protocols performance in different mobility conditions, i.e., varying maximum node speed. The maximum node speed varies from 3 *m/s* to 18 *m/s* and the number of destinations is set to 5, a reasonable scalable figure considering the over all population of 50 nodes.

Figure 4.31 and Figure 4.32 show the result of the ratio of bad frames and the number of bad periods of the all protocols, respectively. We can observe that both the ratio of bad frames and the number of bad periods are almost the same for all of them. Figure 4.33 and Figure 4.34 show that Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR protocols have a lower overhead compared to Serial MDTMR. Again, this is because Serial MDTMR uses a native flooding based tree construction.

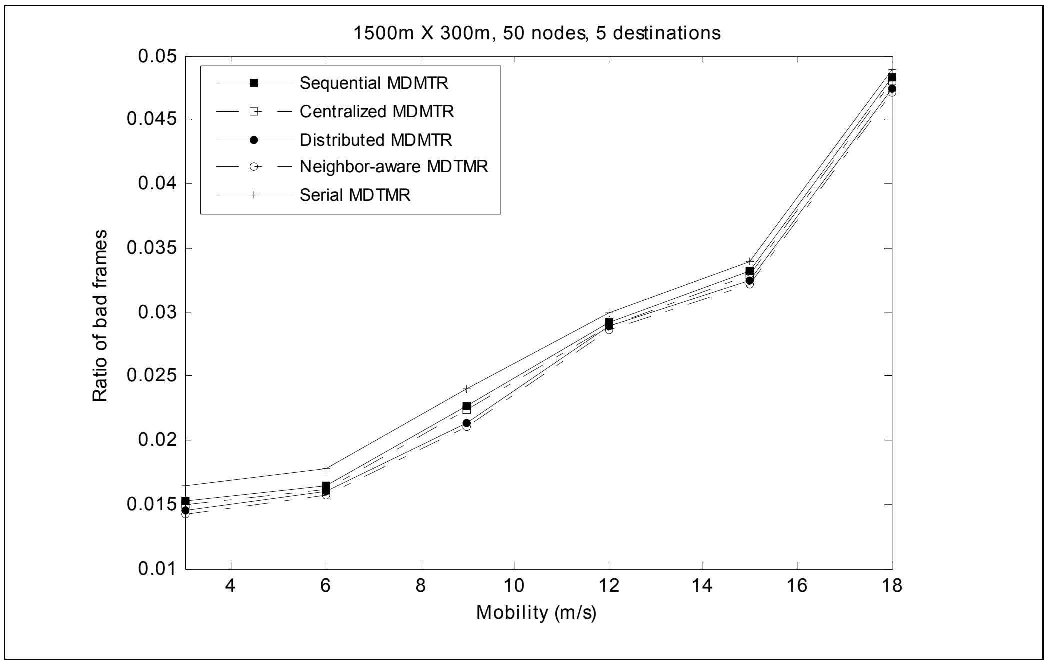


Figure 4.31 Performance evaluation for multiple multicast trees protocols: Ratio of bad frames vs. Node mobility.

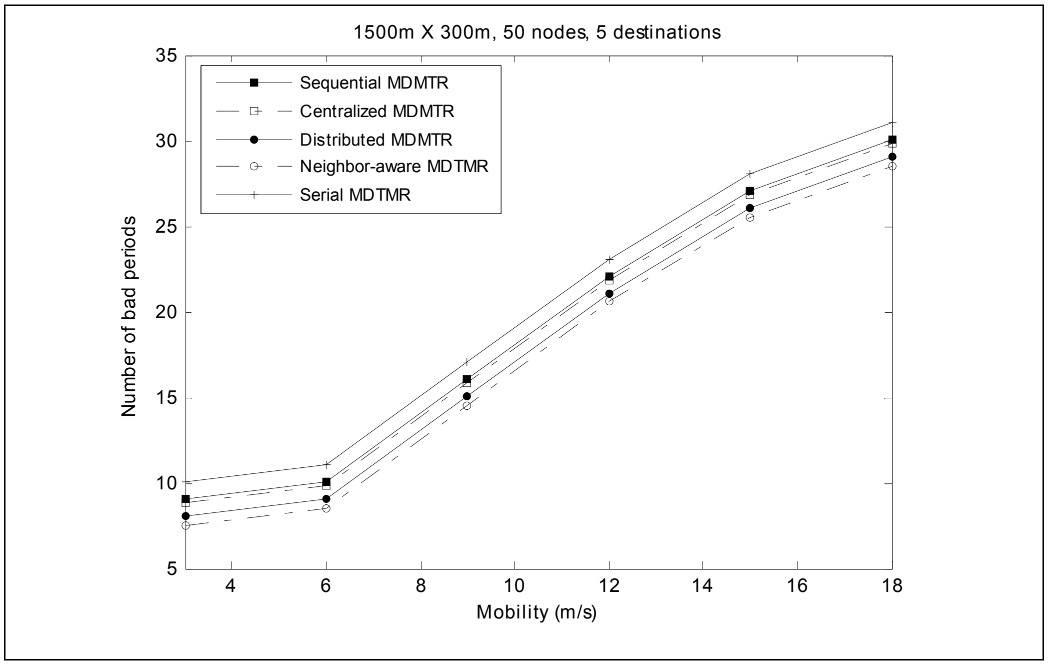


Figure 4.32 Performance evaluation for multiple multicast trees protocols: Number of bad periods vs. Node mobility.

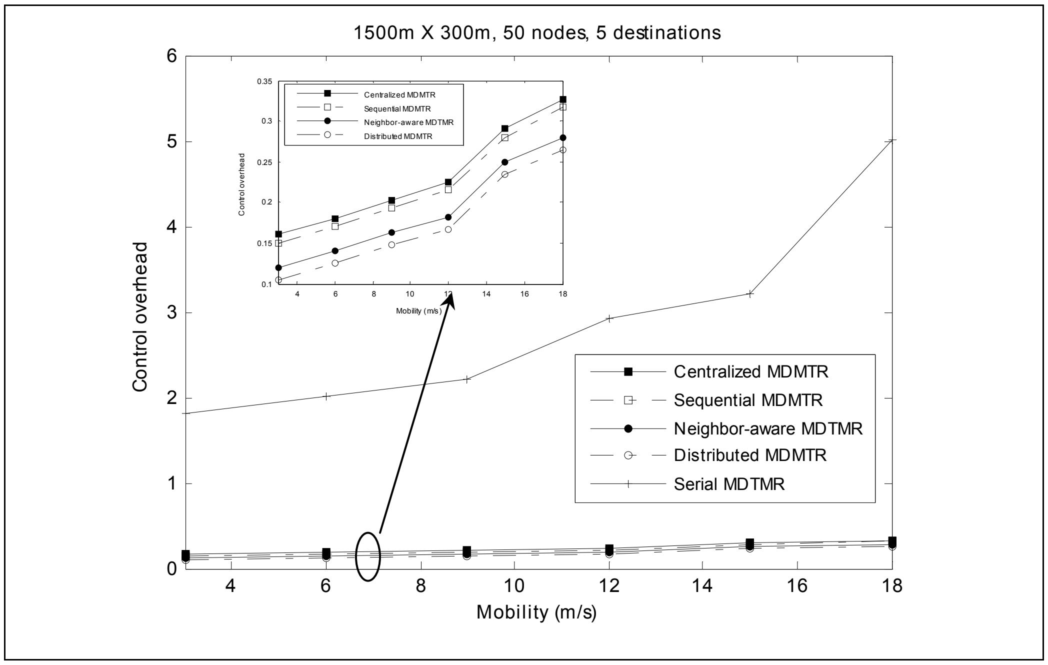


Figure 4.33 Performance evaluation for multiple multicast trees protocols: Control overhead vs. Node mobility.

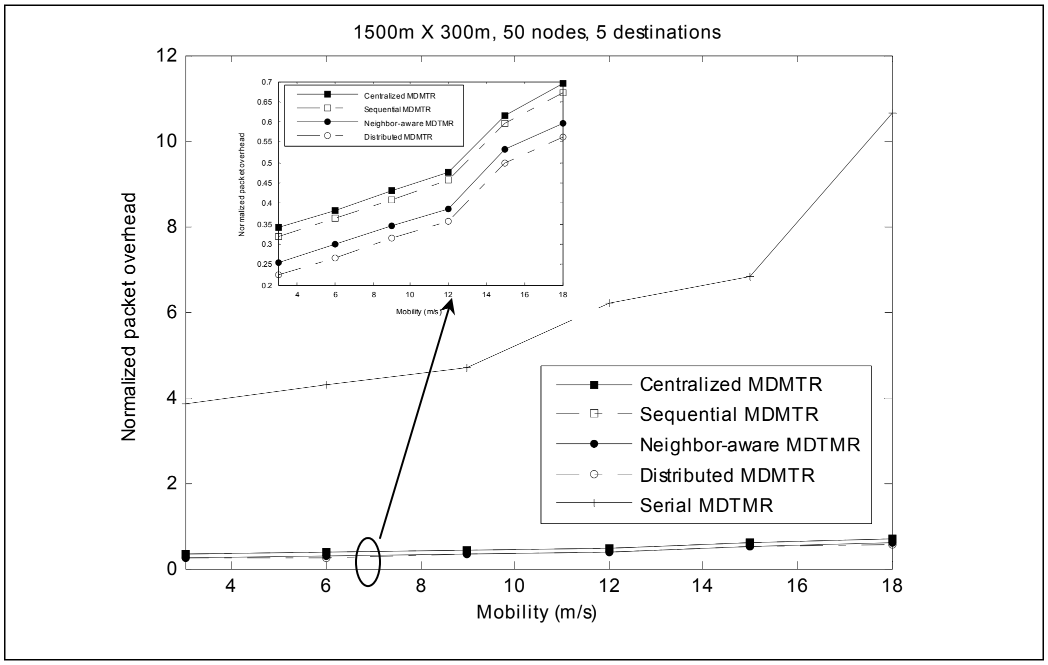


Figure 4.34 Performance evaluation for multiple multicast trees protocols: Normalized packet overhead vs. Node mobility.

4.9 Conclusion

In this chapter, we studied the problem of video multicast for heterogeneous destinations in wireless ad hoc networks. We have proposed four protocols for video multicast improve the user satisfaction. The protocols are Centralized MDMTR, Sequential MDMTR, distributed MDMTR, and Neighbor-aware MDMTR protocols. Centralized MDMTR constructs multiple disjoint multicast trees and assigns MD video in a centralized manner. Furthermore, it deploys the *independent-description* property of MDC. In sequential MDMTR, which is a variant of centralized MDMTR, the assignment of MD video is performed sequentially. On the other hand, Distributed MDMTR and Neighbor-aware MDMTR protocols construct multiple disjoint multicast trees and randomly assign MD video in a distributed fashion. Simulation results showed that our proposed protocols outperformed Serial MDTMR protocols in terms of user satisfaction, overhead, the ratio of bad frames, and the number of bad periods.

CHAPITRE 5

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

In this thesis, we have developed a framework for video streaming over multiple multicast trees in wireless ad hoc networks. Transmitting video streams through multiple trees combats unpredictable packet loss and improves the quality of the received video in wireless ad hoc networks. A video stream can be divided into different *independent* sub-streams (descriptions) using MDC. Our proposed framework can support heterogeneous destinations. Furthermore, it combines the *independent-property* of MDC along with multiple multicast trees to increase the number of assigned (received) video descriptions at each destination node. Thus, the user satisfaction is increased and as a result, the quality of the received video is improved.

In chapter 2, we have presented the general architecture of multipath video streaming over wireless ad hoc networks. Moreover, we have introduced the related works on video multicast over wireless ad hoc networks. The basic idea of these works is to combine the error resilience property of MDC along with multipath to improve the quality of the received video. However, the main objective of these works is to encode the video stream into different *independent* descriptions, using MDC, and transmit each description on different path to the destinations. If any path is broken, packets corresponding to other descriptions can still arrive at the destination node on time. However, existing video multicast protocols developed under the assumption that all destinations wish to receive all the video information sent by the source. In addition they do not take the heterogeneity of destination nodes into consideration. Furthermore, they do not consider the independent-property of MDC.

In chapter 3, we have presented the problem of video description assignment and the construction of multiple node-disjoint multicast trees for a set of heterogeneous destinations. Then we proposed efficient algorithms for this problem. The algorithms are: Serial MDC, Distributed MDC, Centralized MDC, MSPT MDC, and MSMT MDC. Each algorithm adopts different approach to construct multiple multicast trees and to assign MD video. Serial MDC, Distributed MDC, and Centralized MDC algorithms deploy the *independent-description* property of MDC along with multiple multicast trees. Serial MDC and Centralized MDC algorithms construct multiple multicast trees and assign MD video in a centralized manner. However, the main difference between them is how the construction of multicast trees and the assignment of MD video are performed. Serial MDC constructs multiple paths to each destination and assigns each of them different descriptions. Then based on the constructed multiple paths, Serial MDC constructs multiple multicast trees. On the other hand, Centralized MDC first constructs multiple multicast trees and it then assigns different descriptions to each tree. In Distributed MDC the assignment of MD video and the construction of multiple multicast trees are performed in parallel and in a distributed manner. MSPT MDC and MSMT MDC algorithms do not consider the *independent-property* of MDC. Instead, the both algorithms assign MD video sequentially. Both the assignment of MD video and the construction of multiple multicast trees are performed in a centralized manner. We have evaluated the performance of our proposed algorithms through extensive MATLAB simulations. Our simulation results demonstrated the effectiveness of our proposed algorithms. Furthermore, we have showed by simulation that MDC can improve the user satisfaction when compared to LC. In addition, our proposed algorithms have good scalability in terms of number of destinations. Moreover, we have answered our questions about multicast trees construction and MD video assignment and we have showed that the way of multicast trees construction and the MD video assignment can affect the user satisfaction and other metrics (e.g., number of pure forwarders nodes, aggregate tree delay and bandwidth utilization).

In chapter 4, we have studied the problem of video multicast routing for a group of heterogeneous destinations. Then we have proposed four multicast routing protocols for video streaming over wireless ad hoc networks. The protocols are: Centralized MDMTR, Sequential MDMTR, Distributed MDMTR, and Neighbor-aware MDMTR. Centralized MDMTR constructs multiple node-disjoint multicast trees and assigns MD video in a centralized manner. It deploy Centralized MDC algorithm which is proposed in chapter 3. Sequential MDMTR is a variant of Centralized MDMTR protocol. The main difference between them is that Sequential MDMTR assigns MD in a sequential manner, i.e., does not take into consideration the *independent-description* property of MDC. In order to reduce construction delay and overhead, we then proposed Distributed MDMTR and Neighbor-aware MDMTR protocols. Both of them construct multiple node-disjoint multicast trees and assign MD video in a distributed manner. However, the main difference between them is that Neighbor-aware protocol takes into consideration its neighbors' decision before it decides which video description it should select to rebroadcast. This decision is taken during the route request and route reply phases. On the other hand, Distributed MDMTR protocol does not take into consideration its neighbors decisions. Instead, it randomly selects a video description to rebroadcast.

Moreover, our results have demonstrated that proposed protocols achieved higher user satisfaction and lower overhead. In addition, simulation results have showed that our proposed protocols are robust to link failure. Furthermore, they efficiently maintain the multicast group.

5.2 Future Works

Large scale networks: We evaluate our proposed protocols under a small scale networks (50 and 100 nodes). One possible future work is to examine them under a large scale networks (more than 200 nodes). In addition, we assume homogeneous nodes, i.e., they have the same transmission range, processing capabilities, etc. In case of heterogeneous nodes and a large scale networks, can we apply our approaches directly? Or do we need to modify them? For, example, do we need some clustering techniques. More investigations are needed.

Multipoint-to-multipoint transmission: Possible future work includes supporting multicast video transmission using multiple video sources. As commonly known, it is not cost effective to send the same video unicastly to several destinations; therefore, we will investigate the possibility of developing multipoint-to-multipoint transmission over wireless ad hoc networks using disjoint trees. However, in this work, the most challenging part is the establishment of multiple disjoint multicast trees.

Cross-layer design: Finally, we focus, in this work, on the network layer, i.e., the construction of multiple disjoint multicast trees and the assignment of MD video. However, to improve the quality of the received video, the application layer performance metric, i.e., video distortion should be taken into consideration. Therefore, our objective is to jointly optimize routing and the video distortion. The problem formulated for the cross-layer multicast routing is highly complex and is expected to be NP-complete. Therefore, efficient heuristic algorithms would be most useful in practice.

ANNEX I

LIST OF PUBLICATIONS

Journals

Osamah Badarneh, Michel Kadoch, and Ahmed K. Elhakeem. 2008. «QoS multilayered multicast routing protocol for video transmission in heterogeneous wireless ad hoc networks». *WSEAS Transactions on Computers*, Vol. 7, no. 6, p. 680-693.

Osamah Badarneh, and Michel Kadoch. 2009. «Multicast Routing Protocols in Mobile Ad Hoc Networks: A Comparative Survey and Taxonomy». *EURASIP Journal on Wireless Communications and Networking*, Vol. 2009. 42 p.

Conferences

Osamah Badarneh, Yi Qian, Bo Rong, Ahmed Elhakeem, and Kadoch Michel. 2009. «Multiple Description Coding Based Video Multicast over Heterogeneous Wireless Ad Hoc Networks». *In Proc. IEEE ICC*. p. 1-6.

Osamah Badarneh, Michel Kadoch, and Ahmed K. Elhakeem. 2009. «Video Multicast Based Multiple Description Coding and Multi-Paths in Wireless Ad hoc Networks». *In Proc. IEEE CCECE*. p. 604-609.

Osamah Badarneh, Michel Kadoch, and Ahmed K. Elhakeem. 2009. «Supporting Video Multicast in Wireless Ad Hoc Networks Using Multiple Paths and Multiple Description Coding». *In Proc. IEEE ITNG*. p. 646-651.

Osamah Badarneh, Michel Kadoch, and Ahmed K. Elhakeem. 2008. «A new approach for the construction of multiple multicast trees using multiple description video for wireless ad hoc networks». *In Proc. IEEE LCN*, p. 152-159.

Osamah Badarneh, Michel Kadoch, and Ahmed K. Elhakeem. 2008. «Multilayered Video Multiple Trees Multicast Algorithms for Heterogeneous Wireless Ad Hoc Networks». *In Proc. IEEE NCA*, p. 220-223.

Osamah Badarneh, Michel Kadoch, and Ahmed K. Elhakeem. 2008. «Multilayered Multicast Algorithms for Ad Hoc Wireless Networks». *In Proc. WSEAS ACACOS*, p. 586-561.

LISTE OF REFERENCES

Agrawal, D., T. Bheemarjuna Reddy et C. Siva Ram Murthy. 2006. « Robust Demand-Driven Video Multicast over Ad hoc Wireless Networks ». *BROADNET*. p. 1-10.

Anirudh, B., T. Bheemarjuna Reddy et C. Siva Ram Murthy. 2006. « K-tree : A multiple tree video multicast protocol for ad hoc wireless networks ». *in Proc. of HiPC*. p. 424-435.

Apostolopoulos, J. G. 2001. « Reliable video communication over lossy packet networks using multiple state encoding and path diversity ». *Proc. SPIE VCIP*. vol. 4310. p. 392-409.

Asif, A., et M. G. Kouras. 2006. « Scalable Video Codec by Noncausal Prediction,Cascaded Vector Quantization, and Conditional Replenishment ». *IEEE Trans. on Multimedia,* vol. 8, n^o 1, p. 19-31.

Asif, A., U. T. Nguyen et Guohua Xu. 2006. « Scalable Video Multicast over MANETs ». *in IEEE Workshop on Multimedia Signal Processing*. p. 403 - 408.

Badarneh, Osamah, et Michel Kadoch. 2009. « Multicast Routing Protocols in Mobile Ad Hoc Networks: A Comparative Survey and Taxonomy ». *EURASIP Journal on Wireless Communications and Networking*. vol. 2009. 42 p.

Badarneh, Osamah, Michel Kadoch et Ahmed K. Elhakeem. 2008. « A new approach for the construction of multiple multicast trees using multiple description video for wireless ad hoc networks ». *in Proc. of IEEE LCN*. p. 152-159.

Badarneh, Osamah, Michel Kadoch et Ahmed K. Elhakeem. 2009a. « Supporting Video Multicast in Wireless Ad Hoc Networks Using Multiple Paths and Multiple Description Coding ». *in Proc. of IEEE ITNG*. p. 646 - 651.

Badarneh, Osamah, Michel Kadoch et Ahmed K. Elhakeem. 2009b. « Video Multicast Based Multiple Description Coding and Multi-Paths in Wireless Ad hoc Networks ». *in Proc. IEEE CCECE*. p. 604 - 609.

Badarneh, Osamah, Yi Qian, Bo Rong, Ahmed K. Elhakeem et Michel Kadoch. 2009. « Multiple Description Coding Based Video Multicast over Heterogeneous Wireless Ad Hoc Networks ». *in Proc. ICC*. p. 1-10.

Bajaj, S., L. Breslau et S. Shenker. 1998. « Uniform versus priority dropping for layered video ». *ACM SIGCOMM Computer Communication Review,* vol. 28, n^o 4, p. 131-143.

Baker, D. J., et A. Ephremides. 1981. « The architectural organization of a mobile radio network via a distributed algorithm ». *IEEE Trans.Commun.,* vol. 29, p. 1320-1332.

Bonuccelli, A.A. Bertossi and M.A. 1995. « Code Assignment for Hidden Terminal Interference Avoidance in Multihop Packet Radio Networks ». *IEEE/ACM Trans. Networks,* vol. 3, n^o 4, p. 441-449.

Broch, J., D. A. Maltz, D. B. Johnson, Y. C. Hu et J. Jetcheva. 1998. « A performance comparison of multi-hop wireless ad hoc network routing protocols ». *in Proc. ACM/IEEE MobiCom*. p. 85-97.

Chen, J., S.-H. Gary Chan et V. O. K. Li. 2004. « Multipath routing for video delivery over bandwidth-limited networks ». *IEEE Journal on Selected Areas in Communications,* vol. 22, n^o 10, p. 1920-1932.

Chiang, C-C., M. Gerla et L. Zhang. 1998a. « Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks ». *ACM-Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing,* vol. 1, p. 187-196.

Chiang, C.-C., Mario Gerla et Lixia Zhang. 1998b. « Adaptive Shared Tree Multicast in Mobile Wireless Networks ». *in Proc. of IEEE GLOBECOM* (Sydney (Australia), 8-12 Nov. 1998). p. 193-207.

Chiang, Ching-Chuan, et Mario Gerla. 1997. « Routing and multicast in multihop, mobile wireless networks ». *in Proc. of the IEEE International Conference on Universal Personal Communications (ICUPC)* (San Diego (USA), 12-16 Oct 1997). p. 28-33.

Chiang, Ching-Chuan, Mario Gerla et Lixia Zhang. 1997. « Shared tree wireless network multicast ». *in Proc. of the 6th International Conference on Computer Communications and Networks* (Las Vegas (USA), 22-25 Sep. 1997). p. 28-33.

Chlamtac, I., et S. Kutten. 1985. « On broadcasting in radio networks problem analysis and protocol design ». *IEEE Trans. Commun.,* vol. 33, p. 1145-1158.

Chlamtac, I., et S. S. Pinter. 1987a. « Distributed nodes organization algorithm for channel access in a multihop dynamic radio network ». *IEEE Trans. Comput.,* vol. 36, n^o 6, p. 728-737.

Chlamtac, I., et S. S. Pinter. 1987b. « Distributed nodes organization algorithm for channel access in a multihop dynamic radio network ». *IEEE Trans. Comput.,* vol. C-36, n^o 6, p. 728-737.

Chow, C.-O., et H. Ishii. 2008. « Multiple tree multicast ad hoc on demand distance vector (mt-maodv) routing protocol for video multicast over mobile ad hoc networks ». *IEICE-Trans. Commun.,* vol. E91-B, p. 428-436.

Clausen, T., et P. Jacquet. 2003. *Optimized link state routing protocol (OLSR)*. <www.ietf.org/rfc/rfc3626.txt>. 10 Oct. 2009.

Deering, S. 1989. *Host extensions for IP multicasting*. On line. <http://www.ietf.org/rfc/rfc1112.txt>.10 Oct. 2009.

Fantacci, R., A. Ferri et D. Tarchi. 2005. « A MAC Technique for CDMA Based Ad-Hoc Networks ». *in Proc. IEEE WCNC*. Vol. 1, p. 645-650.

Fitzek, F.H.P., B. Can, R. Prasad et M. Katz. 2004. « Overhead and quality measurements for multiple description coding for video services ». *in Proc. WPMC*. Vol. 2, p. 524-528. Italy.

Fitzek, F.H.P., B. Can, R. Prasad et M. Katz. 2005. « Traffic analysis and video quality evaluation ofmultiple description coded video services for fourth generation wireless IP networks ». *Wirel. Pers. Commun.,* vol. 35, p. 187-200.

Fragouli, C., J. Boudec et J. Widmer. 2006. « Network Coding: An Instant Primer ». *in Proc. of ACM SIGCOMM*. Vol. 36, p. 63-68.

Gerla, M., et J. T.-C. Tsai. 1995. « Multicluster, mobile, multimedia radio network ». *ACM-Baltzer J. Wireless Networks,* vol. 1, n° 3, p. 255-265.

Gerla, Mario, Ching-Chuan Chiang et Lixia Zhang. 1999. « Tree Multicast Strategies in Mobile, Multihop Wireless Networks ». *ACM/Baltzer Journal on Mobile Networks and Applications (MONET)*. p. 193-207.

Gogate, N., D.-M. Chung, S. S. Panwar et Y. Wang. 2002. « Supporting image and video applications in a multihop radio environment using path diversity and multiple description coding ». *IEEE transactions on circuits and systems for video technology,* vol. 12, n° 9, p. 777-792.

Goodman, D., R. A. Valenzuela, K. T. Gayliard et B. Ramamurthi. 1989. « Packet reservation multiple access for local wireless communications ». *IEEE Trans. Commun.,* vol. 37, p. 885-890.

Goyal, V. K. 2001a. « Multiple description coding: Compression meets the network ». *IEEE Signal Process Mag.,* vol. 18, p. 74-94.

Goyal, Vivek K. 2001b. « Multiple Description Coding: Compression Meets the Network ». *IEEE Signal Processing Magazine,* vol. 18, n° 5, p. 74-93.

He, Y., I. Lee et L. Guan. 2009. « Optimized Video Multicasting over Wireless Ad Hoc Networks Using Distributed Algorithm ». *IEEE Trans. Circuits Syst. Video Technol.,* vol. 19, n° 6, p. 796-807.

Ho, T., M. Mdard, J. Shi, M. Effros et D. R. Karger. 2006. « On Randomized Network Coding ». *in Proc. of Allerton Conference on Communication, Control, and Computing.* Vol. 36, p. 63-68.

Holbrook, Hugh, et Brad Cain. 2006. *Source-Specific Multicast for IP.* <http://www.isi.edu/in-notes/rfc4607.txt>. 10 ct. 2009.

Holland, G., N. H. Vaidya et P. Bahl. 2001. « A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks ». *in Proc. of ACM MOBICOM.* p. 236-251.

Hu, L. 1993. « Distributed code assignments for CDMA packet radio networks ». *IEEE/ACM Trans. Networking,* vol. 1, n° 6, p. 668-677.

Hung, W.-C., K. L. E. Law et A. Leon-Garcia. 2002. « A dynamic multichannel MAC for ad-hoc LAN ». *in Proc. of Biennial Symposium on Communications.* Vol. 2, p. 377-381.

Ilyas, M. 2002. *Handbook of Ad Hoc Wireless Networks*, 1st ed. Coll. « CRC Press ». 624 p.

Jetcheva, Jorjeta, et David B. Johnson. 2001. « Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks ». *in Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)* (California (USA), 04 - 05 Oct. 2001). p. 33-44.

Klaue, J., B. Rathke et A. Wolisz. 2003. « EvalVid- A framework for video transmission and quality evaluation ». *in Proc. TOOLS.* p. 255-272.

Kompella, V. P., J. C. Pasquale et G. C. Polyzos. 1992. « Multicasting for Multimedia Applications ». *Proc. of IEEE Infocom,* vol. 3, n° 2078-2085.

Kou, L., G. Markowsky et L. Berman. 1981. « A fast algorithm for Steiner trees ». *Acta Informatica,* vol. 15, n° 2, p. 141-145.

Lee, S.J., William Su et Mario Gerla. 2002. « On-demand multicast routing protocol in multihop wireless mobile networks ». *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communications in Wireless Mobile Networks,* vol. 7, n° 6, p. 441-453.

Lee, Sung-Ju, M. Gerla et Ching-Chuan Chiang. 1999. « On-Demand Multicast Routing Protocol ». *in Proc. IEEE WCNC.* p. 1298-1302.

Lin, C. R., et M. Gerla. 1997. « Adaptive clustering for mobile wireless networks ». *IEEE J. Select. Areas Commun.,* vol. 15, p. 1265-1275.

Lin, C. R., et J-S. Liu. 1999. « QoS Routing in Ad Hoc Wireless Networks ». *IEEE Journal on Selected Areas in Communications,* vol. 17, n° 8, p. 1426-1438.

Lin, Chunhung Richard. 2001. « Admission Control in Time-Slotted Multihop Mobile Networks ». *IEEE Journal on Selected Areas in Communications,* vol. 19, n° 10.

Liu, D., Y. Cai et G. Tu. 2006. « Novel Packet Coding Scheme Immune to Packet Collisions for CDMA-Based Wireless Ad Hoc Networks ». *in Proc. IEE Commun.* Vol. 153, p. 1-4.

Makansi, T. 1987. « Transmitter-oriented code assignment for multihop packet radio ». *IEEE Trans. Commun.,* vol. 35, n° 12, p. 1379-1382.

Mao, S., X. Cheng, Y. T. Hou et H. D. Sherali. 2006. « Multiple description video multicast in wireless ad hoc networks ». *ACM/Kluwer Mobile Networks Appl.,* vol. 11, p. 63-73.

Mao, S., S. Lin, S. Panwar, Y. Wang et E. Celebi. 2003. « Video transport over ad hoc networks: multistream coding with multipath transport ». *IEEE J. Sel. Areas Commun.,* vol. 21, n° 10, p. 1721-1737.

Perkins, C. 2000. *Ad Hoc Networking*, 1st ed. Coll. « Addison Wesley ». 384 p.

Perkins, Charles E. 1997. « Ad Hoc On Demand Distance Vector (AODV) Routing ». *Internet-Draft, draft-ietf-manet-aodv-00.txt.*

Royer, E.M., et C.E. Perkins. 2000. *Multicast ad hoc on demand distance vector (maodv) routing.* <Internet-Draft, draft-ietf-draft-maodv-00.txt>.

Software, H.264/AVC Reference. 2009. *Institut Nachrichtentechnik, Heinrich-Hertz-Institut.* <http://iphome.hhi.de/suehring/tml/download/>. 10 Oct. 2009.

Sousa, E. S., et J. A. Silvester. 1988. « Spreading code protocols for distributed sptead-spectmm packet radio net works ». *IEEE Trans. Commu.,* vol. 36, n° 3, p. 272-281.

Tang, X., et A. Zakhor. 2002. « Matching pursuits multiple description coding for wireless video ». *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 12, n° 6, p. 566-575.

Toh, C.-K. 2002. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, 1st ed. Coll. « Prentice Hall ». 336 p.

Wang, Y., et Q. F. Zhu. 1998. « Error control and concealment for video communication: a review ». *Proc. IEEE,* vol. 86, n° 5, p. 974-997.

Wang, Yao, et Shunan Lin. 2002. « Error-resilient video coding using multiple description motion compensation ». *IEEE transactions on circuits and systems for video technology,* vol. 12, n° 6, p. 438-452.

Wei, Wei, et A. Zakhor. 2007. « Multiple Tree Video Multicast Over Wireless Ad Hoc Networks ». *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, p. 2-15

Wei, Wei, et Avideh Zakhor. 2004. « Multipath Unicast and Multicast Video Communication over Wireless Ad Hoc Networks ». *in Proc. BROADNETS*, p. 496-505.

Ye, Z., S. V. Krishnamurthy et S. K. Tripathi. 2003. « A framework for reliable routing in mobile ad hoc networks ». *in Proc. INFOCOM,* vol. 1, p. 270- 280.

Zhang, X. F., et L. Jacob. 2004. « MZRP: An Extension of the Zone Routing Protocol for Multicasting in MANETs ». *Journal of Information Science and Engineering,* vol. 20, n° 3, p. 535-551.

Zhou, X., J. Li et J. Yang. 2005. « A Novel Power Control Algorithm and MAC Protocol for CDMA-Based Mobile Ad Hoc Network ». *in Proc. IEEE MILCOM*. p. 1-7.