# QUALITY-OF-SERVICE-AWARE MULTICAST ROUTING FOR MULTIMEDIA APPLICATIONS IN MOBILE AD HOC NETWORKS

by

Kaan Bür

B.S., Control and Computer Engineering, İstanbul Technical University, 1995

M.S., Computer Engineering, Boğaziçi University, 1998

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2006

QUALITY-OF-SERVICE-AWARE MULTICAST ROUTING FOR
MULTIMEDIA APPLICATIONS IN MOBILE AD HOC NETWORKS

APPROVED BY:

| | |
|---|---|
| Prof. Cem Ersoy<br>(Thesis Supervisor) | ..... |
| Prof. Lale Akarun | ..... |
| Prof. M. Ufuk Çağlayan | ..... |
| Prof. Hakan Deliç | ..... |
| Assoc. Prof. Sema Oktuğ | ..... |

DATE OF APPROVAL .....

# ACKNOWLEDGEMENTS

It has been a decade since my great adventure called “graduate study” began. All this time, I have met many valuable people, whom I have learned a lot from. I hope that the odyssey will continue with new challenges as well as new friendships to develop my academic skills and help me work for the good of society.

First of all, I would like to express my deepest gratitude to Prof. Cem Ersoy, not only for his supervision in this thesis, but also for all his efforts through these years. I will forever be thankful to him for educating me the way he did. I hope I can make him proud as an academician some day. I would also like to thank the members of my thesis jury, Prof. Lale Akarun, Prof. M. Ufuk Çağlayan, Prof. Hakan Deliç and Assoc. Prof. Sema Oktuğ for their time and their valuable comments.

The past and present members of NETLAB contributed to this thesis by being a part of a wonderful, friendly community who shares ideas and fun equally productively. Atay Özgövde and Rabun Koşar, in particular, provided the computational infrastructure throughout the most intensive period of my work. And my former roommate Dr. E. İlker Oyman came up with a bunch of new ideas whenever there was desperate need for some.

Finally, I want to thank my family. It is their encouragement and faith in me that made things bearable. Especially my lovely wife Dilek, who makes me feel like I could reach every star in the sky… She suffered with me through the most difficult stages of this study and supported me unconditionally. After all, it is only the giving that makes you what you are. I will never forget her grace under pressure.

This work was supported in part by the State Planning Organization of Turkey, under grant numbers DPT98K120890 – DPT03K120250 and the university research program of OPNET Technologies Inc, USA.

# ABSTRACT

## QUALITY-OF-SERVICE-AWARE MULTICAST ROUTING FOR MULTIMEDIA APPLICATIONS IN MOBILE AD HOC NETWORKS

The conceptual shift in the expectations of wireless users towards multimedia and group-oriented computing has a significant impact on today's networks in terms of need for mobility, quality of service (QoS) and multicast routing. Mobile ad hoc networks can provide users with these features. However, it is imperative for them to combine QoS and multicast routing strategies in order to utilize the wireless medium efficiently.

This work defines the ad hoc QoS multicast (AQM) routing protocol, which achieves multicast efficiency by tracking the availability of resources for each node within its neighbourhood. Computation of free bandwidth is based on reservations made for ongoing sessions and the requirements reported by the neighbours. The QoS status is announced at session initiation and updated periodically to the extent of QoS provision. Nodes are prevented from applying for membership if there is no QoS path for the session. When nodes wish to join a session with certain service requirements, a three-phase process ensures that the QoS information is updated and used to select the most appropriate routes. The allowed maximum hop count of the session is taken into account in order to satisfy the delay requirements of the multimedia applications. To cope with the continuous nature of streaming multimedia, AQM nodes check the availability of bandwidth within their neighbourhood not only for themselves but within a virtual tunnel of nodes. Objection queries are issued prior to reservation to avoid excessive resource usage due to allocations made by nodes which cannot detect each other directly. A priority queue determines the transmission order of data packets according to their traffic classes to support even those applications with more stringent QoS requirements. AQM evolves the initial multicast tree into a mesh during data flow to improve robustness. New performance metrics are introduced to evaluate the efficiency of AQM regarding the satisfaction level of session members. Simulation results show that, by applying novel QoS management techniques, AQM significantly improves multicast efficiency for members as well as for sessions.

# ÖZET

## GEZGİN TASARSIZ AĞLARDA ÇOĞULORTAM UYGULAMALARI İÇİN SERVİS NİTELİĞİ DESTEKLİ ÇOĞULYAYIN

Telsiz ağlarda kullanıcı beklentilerinin çoğulortam ile gruplar arası bilgisayar kullanımına doğru geçirdiği kavramsal dönüşüm, günümüz ağlarında etkisini gezginlik, servis niteliği, çoğulyayın gereksinimi olarak gösterir. Gezgin tasarsız ağlar kullanıcılara bu olanağı sağlayabilir. Ancak, telsiz iletişim ortamının etkin kullanımı için, çoğulyayın yol atama yordamlarının nitelikli servis ile bütünleştirilmesi temel bir gerekliliktir.

Bu çalışma, çoğulyayın etkinliğini her düğümün kendi komşuluk alanı içindeki ağ kaynaklarının yeterliğini izleyerek sağlayan, tasarsız, servis niteliği destekli bir çoğulyayın yol atama (AQM – ad hoc quality of service multicast routing) yordamı tanımlar. Kullanılabilir bantgenişliği süregelen oturumlara ayrılan ve komşular tarafından bildirilen gereksinim düzeylerine göre belirlenir. Servis nitelik düzeyi oturum açılışında duyurulup, oturumun gerektirdiği nitelik düzeyinin izin verdiği sınırlara dek düzenli olarak güncellenir. Düğümlerin, uygun servis niteliğinin sağlanamayacağı oturumlar için katılım isteğinde bulunmaları engellenir. Bir düğüm bir oturuma katılacağında, üç adımdan oluşan bir süreç gereken niteliklere en uygun yolu seçer. Çoğulortam uygulamalarının gerektirdiği gecikme sınırlarına uymak için, adayların oturum sunucusuna kadar izin verilen sıçrama sayısı sınırlandırılır. Akışkan çoğulortamın doğasına uygun olarak, düğümler bantgenişliği gereksinimlerini yalnız kendileri için değil, verinin sürekli akacağı sanal bir tünel boyunca belirler. Kaynakların aşırı kullanımını önlemek amacıyla, birbirini doğrudan algılamayan düğümlerin olası bir kaynak ayırma işleminden etkilenip etkilenmediği, itiraz sorgularıyla denetlenir. Bir öncelik kuyruğu, veri paketlerinin iletim sırasını ait oldukları trafik sınıfına gore belirler. AQM, gürbüzlüğünü artırmak için, oluşturduğu çoğulyayın ağacını veri akışı sırasında bir örgüye evirir. AQM'nin oturum üyelerine sağladığı tatmin düzeyi, önerilen yeni başarım ölçütleri ile de değerlendirilmiştir. Bilgisayarlı benzetim sonuçları göstermiştir ki, servis niteliği yönetimindeki yenilikçi teknikleri ile AQM, gerek üyeler, gerekse oturumlar için çoğulyayın etkinliğini önemli ölçüde artırmaktadır.

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ..... iii
ABSTRACT ..... iv
ÖZET ..... v
LIST OF FIGURES ..... ix
LIST OF TABLES ..... xvi
LIST OF SYMBOLS/ABBREVIATIONS ..... xvii
1. INTRODUCTION ..... 1
1.1. Quality of Service Systems and Mobile Ad Hoc Networks ..... 4
1.2. Research Overview and Contributions ..... 6
1.3. Thesis Outline ..... 8
2. A BRIEF HISTORY OF MULTICAST IN AD HOC NETWORKS ..... 10
2.1. Classification of Ad Hoc Multicast Routing Protocols ..... 12
2.2. Shared-Tree-Based Multicast Routing Protocols ..... 13
2.2.1. Ad Hoc Multicast Routing Protocol ..... 13
2.2.2. Adaptive Shared-Tree Multicast Routing ..... 13
2.2.3. Ad Hoc Multicast Routing Protocol Utilizing Increasing ID-Numbers ..... 14
2.2.4. Multicast Ad Hoc On-Demand Distance Vector Routing ..... 15
2.2.5. Mobile Multicast Agents ..... 17
2.2.6. Multicast for Ad Hoc Networks with Swarm Intelligence ..... 18
2.3. Source-Tree-Based Multicast Routing Protocols ..... 19
2.3.1. Multicast Core-Extraction Distributed Ad Hoc Routing ..... 19
2.3.2. Bandwidth-Efficient Multicast Routing ..... 20
2.3.3. Differential Destination Multicast Routing ..... 21
2.3.4. Associativity-Based Ad Hoc Multicast Routing ..... 22
2.3.5. Multicast Zone Routing ..... 23
2.3.6. Weight-Based Multicast ..... 23
2.3.7. Preferred-Link-Based Multicast ..... 24
2.4. Mesh-Based Multicast Routing Protocols ..... 26
2.4.1. Forwarding Group Multicast Protocol ..... 26
2.4.2. Core-Assisted Mesh Protocol ..... 27

2.4.3. On-Demand Multicast Routing Protocol ..... 29
2.4.4. Neighbour-Supporting Ad Hoc Multicast Routing Protocol ..... 31
2.4.5. Dynamic Core-Based Multicast Routing Protocol ..... 32
2.4.6. Route-Driven Gossip ..... 33
2.4.7. Protocol for Unified Multicasting Through Announcements ..... 34
2.5. Multicast Optimization Protocols ..... 36
2.5.1. Independent-Tree Ad Hoc Multicast Routing ..... 36
2.5.2. Lantern-Tree-Based Multicast ..... 36
2.5.3. Probabilistic Predictive Multicast Algorithm ..... 37
3. AD HOC QUALITY OF SERVICE MULTICAST ROUTING ..... 39
3.1. Definitions ..... 40
3.2. The Application Module ..... 44
3.2.1. Usage of QoS Classes ..... 44
3.3. The Session Module ..... 45
3.3.1. Session Initiation ..... 47
3.3.2. Session Updates ..... 49
3.3.3. Session Termination ..... 51
3.3.4. Session Losses ..... 53
3.4. The Membership Module ..... 57
3.4.1. Joining a Session ..... 58
3.4.2. Adding Extra Forwarders ..... 64
3.4.3. Leaving a Session ..... 66
3.5. The Network Module ..... 68
3.5.1. Neighbourhood Maintenance ..... 69
3.6. Comments on Implementation ..... 70
4. ESTIMATION OF BANDWIDTH REQUIREMENTS ..... 74
4.1. Resource Allocation ..... 74
4.2. Virtual Tunnel of Bandwidth ..... 76
4.3. Comments and Related Work ..... 82
5. COMPUTATIONAL EXPERIMENTS ..... 84
5.1. Performance Metrics ..... 84
5.1.1. Satisfaction of Session Members ..... 85
5.1.2. Individual QoS Criteria ..... 86

5.1.3. Effects on Network Load ..... 87
5.2. Simulation Settings ..... 87
5.3. Performance Evaluation ..... 91
5.3.1. Satisfaction of Session Members ..... 91
5.3.2. Individual QoS Criteria ..... 97
5.3.3. Effects on Network Load ..... 104
6. ANALYSIS OF THE CONTROL OVERHEAD ..... 110
6.1. Primary Receiver of a Session ..... 111
6.2. Subsequent Receivers ..... 116
6.3. Overhead of a Join Process ..... 124
6.4. Interpretation of Results ..... 131
7. AN IMPROVEMENT IN ADMISSION CONTROL: OBJECTION QUERIES ..... 138
7.1. Overloaded Neighbourhood Problem ..... 139
7.2. Objection Query Mechanism ..... 140
7.3. Modifications on the Membership Module ..... 142
7.4. Performance Evaluation ..... 146
7.4.1. Satisfaction of Session Members ..... 147
7.4.2. Overloaded Sessions and Members ..... 152
7.4.3. Effects on Network Load ..... 154
7.5. Final Remarks ..... 156
8. AN EXTENSION FOR SERVICE DIFFERENTIATION: PRIORITY QUEUES ..... 157
8.1. Background and Motivation ..... 159
8.2. Priority Queuing Based on QoS Classes ..... 161
8.3. Performance Evaluation ..... 164
8.3.1. Effect of Network Density ..... 165
8.3.2. Effect of QoS Class Distribution ..... 168
8.4. Final Remarks ..... 172
9. CONCLUSIONS ..... 174
9.1. Future Research Directions ..... 176
REFERENCES ..... 178

# LIST OF FIGURES

Figure 1.1. A mobile ad hoc network ..... 3

Figure 2.1. Multicast join operation of MAODV ..... 16

Figure 2.2. Multicast connectivity examples in MANSI ..... 18

Figure 2.3. The join protocol of MCEDAR ..... 19

Figure 2.4. Route setup in BEMR ..... 21

Figure 2.5. Examples for the join process of various types of nodes in PLBM ..... 25

Figure 2.6. Forwarding group and tables in FGMP ..... 27

Figure 2.7. Traffic flow from router *h* and non-member source *A* in CAMP ..... 28

Figure 2.8. The forwarding group concept of ODMRP ..... 30

Figure 2.9. Multicast mesh creation in NSMP ..... 31

Figure 2.10. The mesh topology of DCMP ..... 32

Figure 2.11. An example run of RDG ..... 34

Figure 2.12. Mesh creation in PUMA ..... 35

Figure 2.13. A tree, a lantern-tree and a worst-case lantern-tree in LTM ..... 37

Figure 3.1. The architecture of the AQM protocol ..... 40

Figure 3.2. Procedure for the initiation of a new session ..... 47

Figure 3.3. Procedure for handling the received session initiations ..... 48

Figure 3.4. The AQM session initiation process ..... 49

Figure 3.5. Procedure for sending a session update ..... 50

Figure 3.6. Procedure for handling the received session updates ..... 51

Figure 3.7. Procedure for the termination of a session ..... 52

Figure 3.8. Procedure for handling the received session terminations ..... 52

Figure 3.9. Procedure for the announcement of a lost session ..... 54

Figure 3.10. Procedure for handling the received lost session announcements ..... 56

Figure 3.11. Procedure for the request to join a session ..... 59

Figure 3.12. Procedure for the reception of a join request ..... 60

Figure 3.13. Procedure for the reception of a join reply ..... 61

Figure 3.14. Procedure for the reception of a join reserve ..... 63

Figure 3.15. The AQM session joining process ..... 64

Figure 3.16. Procedure for the addition of an extra forwarder during data reception ..... 65

Figure 3.17. Procedure for the addition of an extra receiver during data transmission ..... 66

Figure 3.18. Procedure for leaving a session ..... 67

Figure 3.19. Procedure for handling the received session leave notification ..... 67

Figure 4.1. The virtual tunnel of bandwidth for the multicast members ..... 77

Figure 4.2. The multicast graph of $s_0$ and the neighbourhood of $n_1$ ..... 78

Figure 4.3. The propagation of a join request along the virtual tunnel of bandwidth ..... 80

Figure 5.1. Member QoS sustainability ratio as a function of network density ..... 92

Figure 5.2. Member acceptance ratio as a function of network density ..... 92

Figure 5.3. Member QoS sustainability ratio as a function of multicast traffic load..... 94

Figure 5.4. Member acceptance ratio as a function of multicast traffic load ..... 94

Figure 5.5. Member QoS sustainability ratio as a function of background traffic load ..... 96

Figure 5.6. Member acceptance ratio as a function of background traffic load ..... 96

Figure 5.7. End-to-end data packet delay as a function of network density..... 98

Figure 5.8. Data packet interarrival time as a function of network density..... 98

Figure 5.9. Data packet loss ratio as a function of network density ..... 99

Figure 5.10. End-to-end data packet delay as a function of multicast traffic load ..... 100

Figure 5.11. Data packet interarrival time as a function of multicast traffic load ..... 100

Figure 5.12. Data packet loss ratio as a function of multicast traffic load ..... 101

Figure 5.13. End-to-end data packet delay as a function of background traffic load..... 102

Figure 5.14. Data packet interarrival time as a function of background traffic load ..... 102

Figure 5.15. Data packet loss ratio as a function of background traffic load ..... 103

Figure 5.16. Background traffic efficiency as a function of network density ..... 105

Figure 5.17. Member control overhead as a function of network density ..... 105

Figure 5.18. Background traffic efficiency as a function of multicast traffic load..... 107

Figure 5.19. Member control overhead as a function of multicast traffic load ..... 107

Figure 5.20. Background traffic efficiency as a function of background traffic load ..... 109

Figure 5.21. Member control overhead as a function of background traffic load ..... 109

Figure 6.1. The one-hop neighbourhood of the session server ..... 112

Figure 6.2. The two-hop neighbourhood of the session server ..... 113

Figure 6.3. The multicast tree after the first receiver joins the session ..... 117

Figure 6.4. The coverage area of the multicast tree after the first receiver ..... 117

Figure 6.5. An approximation to the one-hop neighbourhood of the multicast tree..... 121

Figure 6.6. A lower-bound to the one-hop neighbourhood of the multicast tree ..... 122

Figure 6.7. An approximation to the two-hop neighbourhood of the multicast tree ..... 123

Figure 6.8. The propagation of a join request from the requester towards the server ..... 126

Figure 6.9. The areas containing the nodes involved in the join process ..... 127

Figure 6.10. Integral boundaries of the areas involved in the join process ..... 127

Figure 6.11. Approximations to the area involved in the join process ..... 130

Figure 6.12. Hop count probabilities for the first and second receivers ..... 132

Figure 6.13. Probability distribution of the hop count for the first and second receivers 132

Figure 6.14. Expected value of the hop count for the first and second receivers ..... 134

Figure 6.15. Expected value of the hop count as a function of increasing $\gamma$ ..... 134

Figure 6.16. The number of nodes involved in the join process of the first receiver ..... 136

Figure 6.17. The number of nodes involved in the join process of the second receiver .. 136

Figure 7.1. The overload problem in the neighbourhood of $n_6$ ..... 139

Figure 7.2. The objection query received and responded by $n_6$ ..... 141

Figure 7.3. Revised procedure for the reception of a join request ..... 143

Figure 7.4. Revised procedure for the reception of a join reply ..... 144

Figure 7.5. Procedure for the reception of an objection query ..... 145

Figure 7.6. Procedure for the reception of an objection in response to a query ..... 145

Figure 7.7. Procedure for the completion of the query with the objection timeout ..... 145

Figure 7.8. Member QoS sustainability as a function of network density ..... 148

Figure 7.9. Member acceptance ratio as a function of network density ..... 148

Figure 7.10. Class 4 member acceptance ratio as a function of network density ..... 150

Figure 7.11. Class 1-2-3 member acceptance ratio as a function of network density ..... 150

Figure 7.12. Class 4 member QoS sustainability as a function of network density ..... 151

Figure 7.13. Class 1-2-3 member QoS sustainability as a function of network density... 151

Figure 7.14. Member overload ratio as a function of network density..... 153

Figure 7.15. Session overload ratio as a function of network density ..... 153

Figure 7.16. Background traffic efficiency as a function of network density ..... 155

Figure 7.17. Member control overhead as a function of network density ..... 155

Figure 8.1. Extended architecture of the AQM protocol ..... 161

Figure 8.2. Class 1 data packet delay as a function of network density ..... 165

Figure 8.3. Class 2 data packet delay as a function of network density ..... 166

Figure 8.4. Class 3 data packet delay as a function of network density ..... 166

Figure 8.5. Class 4 data packet delay as a function of network density ..... 167

Figure 8.6. Background data packet delay as a function of network density ..... 168

Figure 8.7. Class 1 data packet delay as a function of QoS class distribution ..... 169

Figure 8.8. Class 2 data packet delay as a function of QoS class distribution ..... 169

Figure 8.9. Class 3 data packet delay as a function of QoS class distribution ..... 170

Figure 8.10. Class 4 data packet delay as a function of QoS class distribution ..... 171

Figure 8.11. Background data packet delay as a function of QoS class distribution..... 171

# LIST OF TABLES

Table 2.1. Chronology of ad hoc multicast routing protocols ..... 11

Table 3.1. Definitions for the node states in AQM ..... 42

Table 3.2. Definitions for the tables in AQM ..... 42

Table 3.3. Definitions for the messages in AQM ..... 43

Table 3.4. Data fields used in the table of sessions (TBL_SESSION) ..... 45

Table 3.5. Data fields used in the table of members (TBL_MEMBER) ..... 46

Table 3.6. Data fields used in the table of requests (TBL_REQUEST) ..... 58

Table 3.7. Data fields used in the table of neighbours (TBL_NEIGHBOUR) ..... 68

Table 5.1. QoS requirements of application classes ..... 89

Table 5.2. Simulation parameters for the performance evaluation ..... 90

Table 5.3. Simulation variables of the performance evaluation ..... 90

Table 6.1. Simulation settings for the analysis of the join process ..... 135

Table 8.1. Simulation variables of the queuing performance evaluation ..... 165

# LIST OF SYMBOLS/ABBREVIATIONS

| | |
|---|---|
| $A_f$ | Partial coverage area of an intermediate node on the tree |
| $A_h$ | $h^{\text{th}}$ propagation area between $s$ and $m_l$ in the join process of $m_l$ |
| $A_J$ | Total propagation area between $s$ and $m_l$ in the join process of $m_l$ |
| $A'_J$ | Approximation to $A_J$ |
| $A''_J$ | Alternate approximation to $A_J$ |
| $A_m$ | Partial coverage area of a receiver, a leaf on the tree |
| $A_{Member}$ | Member acceptance ratio |
| $A_s$ | Partial coverage area of a server, the root of the tree |
| $A_{T,h}$ | $h$-hop coverage area of the multicast tree |
| $A'_{T,h}$ | Approximation to $A_{T,h}$ |
| $a$ | Number of accepted members, i.e., receivers |
| $b_i$ | Bandwidth requirement of node $i$ |
| $b_{N_i}$ | Bandwidth allocation in the neighbourhood of node $i$ |
| $b_{s_k}$ | Bandwidth requirement of a session $s_k$ served by node $k$ |
| $C_{Member}$ | Member control overhead |
| $c$ | Number of control packets processed |
| $d$ | Number of members dropped off a session due to insufficient QoS |
| $d_{ij}$ | Euclidian distance between two nodes $i$ and $j$ |
| $E[H_l]$ | Expected value of $H_l$ |
| $F$ | Set of forwarders in a session |
| $F_{i,s_k}$ | Set of downstream forwarders of $s_k$ in the neighbourhood of node $i$ |
| $f$ | Number of session forwarders |
| $f_h$ | A node from the set $F$ |
| $g$ | Number of join requests |
| $H_l$ | Discrete random variable for the probability $P\{H_l = h\}$ |
| $h$ | Variable for the number of hops that $m_l$ needs to reach $s$ |
| $h_l$ | Hop distance between $m_l$ and $s$ |
| $M$ | Set of the members of a session |
| $m_l$ | $l^{\text{th}}$ member of a session |

| | |
|---|---|
| $N$ | Set of nodes in the ad hoc network |
| $N_i$ | Set of nodes in the neighbourhood of node $i$ |
| $n_i$ | $i^{\text{th}}$ node in the ad hoc network |
| $O_{Member}$ | Member overload ratio |
| $O_{Session}$ | Session overload ratio |
| $o$ | Number of overloaded members |
| $P\{H_l = h\}$ | Probability that $m_l$ reaches $s$ in $h$ hops |
| $P'\{H_l = h\}$ | Approximation to $P\{H_l = h\}$ |
| $P''\{H_l = h\}$ | Lower-bound approximation to $P\{H_l = h\}$ |
| $Q_{Member}$ | Member QoS sustainability ratio |
| $q_{i,s_k}$ | QoS bandwidth necessary in the neighbourhood of node $i$ for $s_k$ |
| $R$ | Radius of the circular network area |
| $r$ | Transmission range of the nodes in the ad hoc network |
| $\bar{r}$ | Average distance between two consecutive nodes on a tree |
| $S$ | Set of multicast sessions |
| $S_i$ | Set of multicast sessions having node $i$ on their session tree |
| $s$ | Session server located at the centre of the ad hoc network |
| $s_k$ | Session served by node $k$ |
| $T$ | Set of all nodes in a session, the union of the sets $M$ and $F$ |
| $t_i$ | Relative time index of packets arriving at a node |
| $w_{m,h}$ | $h^{\text{th}}$ propagation wave of a join request packet from the receiver |
| $w_{s,h}$ | $h^{\text{th}}$ propagation wave of an initiation packet from the server |
| $y$ | Number of sessions with at least one overloaded member |
| $z$ | Number of servers |
| $\beta$ | Maximum available neighbourhood bandwidth |
| $\beta_i$ | Available bandwidth in the neighbourhood of node $i$ |
| $\phi_{i,s_k}$ | Cardinality of $F_{i,s_k}$ |
| $\gamma$ | Maximum number of hops in the ad hoc network |
| $\mu_J$ | Number of control messages processed during a join process |
| $\nu$ | Cardinality of $N$ |
| $\nu_J$ | Number of nodes within the propagation area of a join request |
| $\rho$ | Node density of the ad hoc network |

| | |
|---|---|
| ABAM | Associativity-based ad hoc multicast |
| ABR | Associativity-based routing |
| AMRIS | Ad hoc multicast routing protocol utilizing increasing ID-numbers |
| AMRoute | Ad hoc multicast routing |
| AODV | Ad hoc on demand distance vector routing protocol |
| AQM | Ad hoc quality of service multicast |
| AQOR | ad hoc QoS on-demand routing |
| ASTM | Adaptive shared-tree multicast |
| BEMR | Bandwidth-efficient multicast routing |
| CACP | Contention-aware admission control protocol |
| CAMP | Core-assisted mesh protocol |
| CDMA | Code division multiple access |
| CEDAR | Core-extraction distributed ad hoc routing |
| CTS | Clear-to-send |
| DCMP | Dynamic core-based multicast routing protocol |
| DDM | Differential destination multicast |
| DiffServ | Differentiated services |
| DSR | Dynamic source routing |
| DVMRP | Distance vector multicast routing protocol |
| FGMP | Forwarding group multicast protocol |
| FIFO | First-in-first-out |
| H-PFQ | Hierarchical packet fair queuing |
| IGMP | Internet group management protocol |
| INSIGNIA | In-band signalling QoS framework |
| IntServ | Integrated services |
| IP | Internet protocol |
| ITAMAR | Independent-tree ad hoc multicast routing |
| JOIN_ERR | Message for *reservation errors* in AQM |
| JOIN_OBJ | Message for *objection queries* in AQM |
| JOIN_REP | Message for *join replies* in AQM |
| JOIN_REQ | Message for *join requests* in AQM |
| JOIN_RES | Message for *join reservations* in AQM |

| | |
|---|---|
| LAN | Local area network |
| LTM | Lantern-tree-based QoS multicast |
| MAC | Medium access control |
| MANSI | Multicast for ad hoc networks with swarm intelligence |
| MAODV | Multicast ad hoc on demand distance vector routing protocol |
| MCEDAR | Multicast core-extraction distributed ad hoc routing |
| MCN_FRCV | Multicast node state *forwarding receiver* in AQM |
| MCN_FWD | Multicast node state *forwarder* in AQM |
| MCN_INIT | Multicast node state *session initiator* in AQM |
| MCN_PRED | Multicast node state *predecessor* in AQM |
| MCN_RCV | Multicast node state *receiver* in AQM |
| MCN_SRV | Multicast node state *session server* in AQM |
| MLD | Multicast listener discovery |
| MMA | Mobile multicast agents |
| MZR | Multicast zone routing |
| NBR_HELLO | Message for *neighbour greetings* in AQM |
| NSMP | Neighbour-supporting multicast protocol |
| ODMRP | On-demand multicast routing protocol |
| PLBM | Preferred-link-based multicast |
| PLBR | preferred-link-based routing |
| PPMA | Probabilistic predictive multicast algorithm |
| PRTMAC | Proactive real-time medium access control |
| PUMA | Protocol for unified multicasting through announcements |
| QoS | Quality of service |
| QPART | QoS protocol for ad hoc real-time traffic |
| RDG | Route-driven gossip |
| RSVP | Resource reservation protocol |
| RTS | Request-to-send |
| SES_INIT | Message for *session initiations* in AQM |
| SES_LEAVE | Message for *session leaving* in AQM |
| SES_LOST | Message for *lost sessions* in AQM |
| SES_TERMINATE | Message for *session terminations* in AQM |

| | |
|---|---|
| SES_UPDATE | Message for *session updates* in AQM |
| SWAN | Stateless wireless ad hoc networks |
| TBL_NEIGHBOUR | Table of *neighbours* in AQM |
| TBL_MEMBER | Table of *session members* in AQM |
| TBL_REQUEST | Table of *join requests* in AQM |
| TBL_SESSION | Table of *sessions* in AQM |
| TCP | Transmission control protocol |
| TDMA | Time division multiple access |
| TFTP | Trivial file transfer protocol |
| TTL | Time-to-live |
| WBM | Weight-based multicast |
| WFQ | Weighted fair queuing |
| WLAN | Wireless local area network |
| ZRP | Zone routing protocol |

# 1. INTRODUCTION

The increasing popularity of video, voice and data communications over the Internet and the rapid penetration of mobile telephony have stimulated a change in consumers' expectations. Even though voice still accounts for the significant part of the world's mobile communications traffic, the number of group-oriented services and multimedia applications is increasing. The evolution of wireless communication technologies has reached a point where it is both popular and easy to integrate them to portable computing devices, which have initially been intended for personal use. Today, a new generation of such computers is being developed, offering users more computational power than ever, in addition to mobility, the principal distinguishing characteristic of personal communication.

Research and development are taking place to define the next generation of wireless broadband multimedia communication systems. The global multimedia network of the future will probably consist of a fixed network with a wired backbone, an infrastructured mobile network with base stations and, at the peripherals, ad hoc mobile networks, which will be connected to the main internetwork via ad hoc switches [1, 2]. While current communication systems are primarily designed for one specific type of application such as speech, video or data, the next generation will integrate various functions and applications incorporating data, audio, graphics, video, images and animation. Wireless communication will provide high-speed, high-quality information exchange between handheld devices. Therefore, it is essential that wireless and multimedia be brought together [3, 4].

The commercial success of the portable computers shows that consumers want their information to be part of themselves. On the other hand, they also want to communicate with others as well as a variety of information services [5]. It is therefore reasonable to assume that people want the same communication capabilities on the move as they enjoy in their home or office [4]. The simultaneous popularity of portable computing and networking poses a paradox, since portable devices are not connected to the conventional wired networks. This paradox can only be resolved by wireless data networks, through which the users retain the advantages of mobility and being connected at the same time [5]. However, the integration of new technologies into these devices requires configuration

like computers, which is a challenge for the inexperienced user. Thus, it becomes an increasingly important feature that, once a mobile device is operational, it is able to configure itself with all its personal and networking capabilities, asking its users only for their personal preferences. Only by making the administrative work transparent to the end user can wireless technologies contribute to the penetration of mobile computing devices.

The widespread use of mobile and handheld computing devices increases the popularity of mobile ad hoc networks, which are self-organizing communication groups formed impromptu by wireless mobile hosts. They make their administrative decisions in a distributed manner without any centralized control. They are free from the boundaries of any pre-existing infrastructure and can be deployed anytime, anywhere [1, 6-8]. The nodes in a mobile ad hoc network move arbitrarily. Thus, the network topology changes frequently and unpredictably. The routing functionality possessed by the nodes enables them to communicate with each other through multihop paths made of intermediate nodes that relay the packets from the source towards the destination, even if these reside beyond the transmission range of each other [1, 6, 9]. Due to their quick and economically less demanding deployment, mobile ad hoc networks are considered for many commercial applications, including home networks, nomadic computing, wireless local area, mesh or sensor networks and an increasing number of collaborative and distributed applications such as short-term communication for emergency operations, search and rescue, disaster relief, public events, game playing and temporary offices [4, 10, 11]. The requirement of a temporary network for instantaneous communication among a group of people makes mobile ad hoc networks an excellent solution for these cases. A major factor that favours them for such tasks is the self-configuration ability of the system with minimal overhead. Figure 1.1 shows an example for a mobile ad hoc network of ten nodes.

There are many applications of mobile ad hoc networks that involve point-to-multipoint or multipoint-to-multipoint communication patterns, which makes the efficient support of group communications a critical issue. The multicast communications model can facilitate effective and collaborative communication among groups [12]. Multicast routing is a promising technique to provide a subset of network nodes with the group-oriented service they demand while not jeopardizing the resource requirements of others. This is important for mobile ad hoc networks. The advantage of multicast routing is that

packets are only replicated when it is necessary to reach two or more receivers on disjoint paths. This way, bandwidth consumption, router processing and delivery delay can be minimized. In wireless networks, it is particularly important to reduce transmission overhead. Thus, multicast routing can improve wireless link efficiency by exploiting the inherent broadcast property of the wireless medium [13]. Combining the features of mobile ad hoc networks with the usefulness of multicast routing, a number of group-oriented applications with close collaborative efforts can be realized [14]. However, node mobility, with constraints of delay, loss and bandwidth, makes multicast routing very challenging. Robustness, efficiency, control overhead, quality of service (QoS) and group management are some of the major issues in designing multicast routing protocols [11].

The emergence of all these new technologies ensures that multimedia applications will dominate the mobile communications usage [15]. In order to meet the qualitative expectations of mobile users for such applications, mobile ad hoc networks need support for multimedia, which can generally be defined as a combination of data, audio, graphics, video, images and animation, showing real-time, variable bit-rate traffic characteristics. This makes QoS a fundamental requirement. The QoS concept defines a guarantee given by the network to satisfy a set of predetermined service performance constraints for the user in terms of end-to-end delay, jitter, available bandwidth and packet loss probability [8] such that the required functionality of an application can be achieved. QoS support for multimedia applications often requires negotiation between the host and the network and is closely related to resource allocation schemes, priority scheduling and call admission control [11].

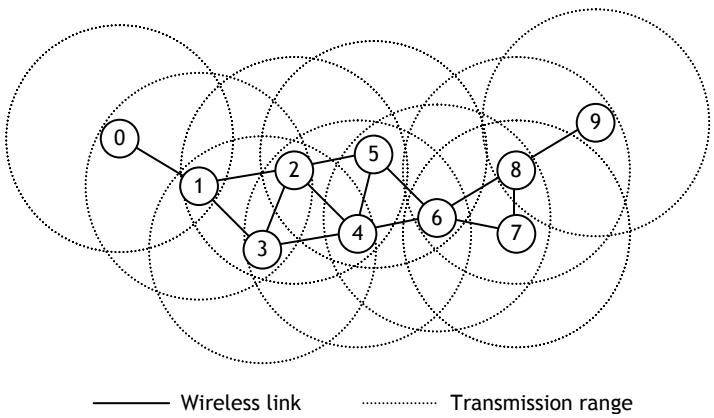



Figure 1.1. A mobile ad hoc network

### 1.1. Quality of Service Systems and Mobile Ad Hoc Networks

A QoS system consists of several components, including service differentiation, admission control and resource allocation [16-18]. Service differentiation schemes use QoS techniques such as priority assignment and fair scheduling. Priority assignment mechanisms are also referred to as packet-level scheduling systems [16]. Every node defines a queuing discipline that changes the waiting times of the frames and assigns smaller values to high-priority traffic to determine which packet to send during the next transmission period [17]. Fair scheduling algorithms, on the other hand, require the cooperation of all the nodes within a neighbourhood to determine which nodes have channel access priority [16]. They partition resources among flows in proportion to a given weight and regulate the waiting times for fairness among traffic classes [17].

Since the exact condition of the wireless network is not known, an accurate decision is not possible regarding the admission of a new flow. Measurement-based admission control mechanisms are based on observations on the existing network status, whereas calculation-based mechanisms make use of performance metrics they define for evaluating the status of the network. Without admission control and bandwidth reservation, the provision of QoS only by differentiating flows and coordinating channel access order is not effective for high traffic loads [18]. A contention-aware admission control protocol (CACP) introduces the concept of an extended contention area covering the carrier sensing range of a node [19]. Admission decisions are based on the information collected from the neighbours in the contention area, which consists of the smallest local bandwidth available and the consumed bandwidth within that area. None of the nodes intentionally breaks QoS by admitting too many flows.

Due to the dynamic nature of mobile ad hoc networks characterized by variable link behaviour, node movements and topology changes, it is very important to design efficient methods of conserving the scarce resources [10]. Once a route is selected for a specific connection request and reservation of resources is completed, these resources are not available to subsequent requests until the end of the granted connection. The objective of resource allocation is to decide how to reserve resources such that QoS requirements of all the applications can be satisfied [20].

Another important feature of a QoS system is the congestion control scheme. Congestion occurs when the data sent exceeds the capacity of the network and causes excessive delay and loss. It can be avoided by predicting it and reducing the transmission rate. If congestion is local, it can also be handled by routing around the congested node without reducing the data rate [21]. A multicast congestion control scheme for multi-layer data traffic is proposed to be applied at the bottlenecks of the multicast tree using the queue states [22]. Some flow information is maintained at each node and data layers are blocked and released to solve congestion problems and adjust the bandwidth rate.

Current research on the implementation of QoS to mobile ad hoc networks is mainly limited to medium access control (MAC) and routing. Generally, a time division multiple access (TDMA) or a clustered code division multiple access (CDMA) over TDMA network synchronized on a frame and slot basis is assumed, where topologies do not change very fast and slot assignment is left to the underlying MAC layer [23-28]. There is a control phase in each frame, whereby nodes exchange connectivity information while clusterheads synchronize slots and frames, in addition to assigning slots and code to connection requests [23]. Each node broadcasts its QoS information during the control phase, at the end of which each node knows the channel reservation status of the next information phase [24]. The goal of the QoS routing algorithm is to find a shortest path such that the available bandwidth on the path is above the minimal requirement. To compute this path, not only the available bandwidth on each link on the path has to be known, but also the scheduling of free slots has to be determined. Thus, heuristics are developed for bandwidth calculation, slot assignment and rerouting in the presence of broken paths [25]. A set of free and non-conflicting slots is calculated on three adjacent links and propagated towards the destination [26]. Bandwidth calculation is done end-to-end; i.e., only destinations reply to connection requests. A timer is refreshed or expired based on the usage of the QoS routes. Based on the free slot information in the path-searching packets they receive, destinations can either select single paths, or determine a multi-path route to satisfy their QoS requirements [27]. Imprecise QoS information is kept at each node for every other, whereas the state of immediate neighbours is traced more accurately [28]. Additional MAC assumptions include a mechanism of beacons, contention resolution and local message broadcasting.

Mobile ad hoc networks possess various unique properties which make them very different from traditional wired and even wireless systems. It is a significant technical challenge to provide reliable high-speed end-to-end communications in mobile ad hoc networks, due to their dynamic topology, distributed management and multihop connections [4]. In addition, the actual throughput of wireless communications is often much less than the maximum radio transmission rate, due to the effects of multiple access, fading, noise and interference conditions. Furthermore, these effects result in time-varying channel capacity, making it difficult to determine the aggregate bandwidth between two endpoints. Finally, resources such as energy, bandwidth, processing power and memory, which are relatively abundant in wired environments, are strictly limited and have to be preserved in mobile ad hoc networks [16].

It is not an easy task to incorporate QoS to ad hoc multicast routing. Wireline QoS algorithms rely on the availability of precise state information, whereas in an ad hoc network this information is inherently imprecise [29]. Nodes join, leave and rejoin the network at any place. Links appear or disappear at any time. Thus, protocols designed for wired networks are not appropriate for ad hoc networks due to their lack of adaptation to the unpredictable network topology and excessive overhead [4, 11]. Various protocols are developed to build and maintain a multicast graph and perform routing in mobile ad hoc networks. However, they rarely attempt to cover the multimedia QoS requirements of sessions within the task of ad hoc multicast routing, which is becoming increasingly important as the demand for mobile multimedia increases. Incremental changes on existing ad hoc schemes cannot efficiently address the critical issues mentioned above.

### 1.2. Research Overview and Contributions

In this thesis, the ad hoc QoS multicast (AQM) routing protocol is presented as a composite solution to the problem. AQM tracks QoS availability within each node's neighbourhood based on current reservations made for ongoing sessions and the requirements reported by the neighbours and announces it at session initiation. This information is updated periodically to the extent of QoS provision. Nodes are prevented from applying for membership if there is no QoS path for the session. When a node wants to join a session with certain service requirements, a request-reply-reserve process ensures

that this QoS information is updated and used to select one of the routes which can meet the requirements of that session. The allowed maximum hop count of the session is taken into account in order to satisfy the delay requirements of the multimedia applications. Objection queries are utilized during this process to avoid excessive allocation of resources. At each node, a priority queue determines the transmission order of data packets according to their traffic classes in order to fulfil even the requirements of those applications with more stringent QoS restrictions. AQM evolves the initial multicast tree into a mesh during data flow to improve robustness. Simulation results show that AQM significantly improves multicast routing efficiency for members and sessions through QoS management. The main contributions of the thesis can be summarized as follows:

- The concept of QoS-awareness is integrated to multicast routing in mobile ad hoc networks. Specifically, the hybrid approach of proactive session management and reactive membership management with the introduction of the objection query mechanism keeps nodes informed on QoS. A simple delay component is added to the protocol by limiting the number of hops allowed to join a session of a specific QoS class. Using these features, AQM nodes are able to make their decisions on sending join requests or replies based on the availability of QoS. Although there are several resource management schemes developed to serve ad hoc routing and multicast protocols as QoS modules, there has not been an ad hoc multicast protocol that incorporates QoS directly in its admission and routing decisions prior to AQM.

- The concept of the virtual tunnel of bandwidth is introduced to compare a node's usable bandwidth to the bandwidth requirement of a new session more accurately. With the help of this concept, the continuous nature of data flow in a multimedia application is taken into account. Thus, the estimation of the available bandwidth is improved, which also helps the nodes make more accurate reservation decisions.

- A priority queue is implemented at each node in order to schedule data packets with regard to the application class that they belong to. Such a queuing discipline is a very important part of a QoS-aware routing protocol since it is the only means of service differentiation during data flow. Thus, AQM can sort data traffic according to their QoS requirements and prioritize flows that are more sensitive to loss and delay.

- The analysis of the control overhead, which considers the multi-hop nature, the limited transmission range and the omni-directional communication property of ad hoc networks, provides a method for the computation of the probabilities that a node reaches a multicast tree in a certain number of hops and an estimate of the number of nodes involved in the process. Some of the results of the analysis are not specific to AQM and can be useful for a general interpretation of ad hoc multicast strategies.

- New performance metrics, which are both qualitative and measurable, are defined in order to evaluate the efficiency of AQM. Since the motivation of the proposed protocol is the provision of QoS to the multicast users in the ad hoc network, it is necessary to define new criteria to measure service satisfaction of the session members. Thus, the concept of QoS sustainability is introduced to evaluate AQM with regard to members with insufficient perceived QoS, which has a direct impact on service satisfaction, the primary QoS criterion. In addition, overloaded members and sessions are observed to examine the efficiency of AQM in resource allocation.

- A mobile ad hoc network environment is designed which supports multiple QoS classes simultaneously, such that a realistic usage scenario can be achieved. Background data traffic is also incorporated in this scenario along with multicast data traffic in order to observe their mutual effects on the performance of each other. Simulations are conducted in this environment for a network lifetime as long as one hour in order to get an impression of the long-term behaviour of AQM maintaining multiple multicast sessions of arbitrary numbers of members in a distributed manner. Since these sessions belong to different service classes, they also have different QoS requirements. Previous performance evaluations in the research literature mainly limit themselves to one multicast session, a restricted number of members and at most a few minutes of simulated time.

### 1.3. Thesis Outline

Section 2 summarizes previous research efforts made in the field of multicast routing in mobile ad hoc networks. A general classification of the protocols is made. Several proposed multicast protocols are examined and their main ideas are briefly described.

Section 3 introduces AQM. First, the modular architecture of the protocol is described. Then the definitions are given for various information entities used. Finally, the procedures used by each module are explained in detail. The algorithmic structures for the initiation and termination of sessions, the joining and leaving of members are presented.

Section 4 describes bandwidth estimation in AQM. It defines a node's bandwidth requirement and a neighbourhood's allocated resources. Then the available bandwidth is estimated. Finally, the term virtual tunnel of bandwidth is introduced, which is a decision mechanism to check more accurately if the available bandwidth is enough to support QoS.

Section 5 evaluates the performance of AQM through simulation. First, the metrics of performance are defined. Then the simulation settings are summarized. Finally, evaluation results are presented. The behaviour of AQM is simulated using different parameters, evaluating it with regard to node density, network load and QoS requirements.

Section 6 analyses the control overhead incurred by AQM. Based on the ranges that ad hoc nodes are able to reach each other and the geometric properties of their intersection, the number of nodes involved in the control processes of AQM is derived. Following this analysis, the overhead of AQM control packets on the ad hoc network is formulated.

Section 7 makes improvements in AQM based on the previously presented results. The problem experienced by the protocol is identified and the objection query mechanism is developed as an extension to AQM in order to improve its performance. The simulations are repeated for the improved version of AQM and the results are interpreted.

Section 8 presents a priority queuing system incorporated in AQM for the timely and successful delivery of data packets based on their traffic classes. The need for service differentiation is explained and the queuing discipline is developed. The simulations are repeated to compare AQM's performance with and without the class-based priority queue.

Section 9 concludes the thesis, summarizing the work done and the results achieved. The contributions are restated and some insight into future research directions is given.

# 2. A BRIEF HISTORY OF MULTICAST IN AD HOC NETWORKS

According to its general definition in the domain of conventional, wired networks, multicast routing is a transmission technique that allows a single data packet sent by a source to be replicated by intermediate nodes at each separation point of disjoint paths as necessary and passed to a selected subset of all possible destinations [30, 31]. When an application has to send the same information to a number of destinations, multicast routing provides a more efficient solution than unicasting the data packets to each node separately or broadcasting them throughout the network. It reduces the transmission overhead both on the source as well as on the network and speeds up the delivery of information at the destinations [32].

The first multicast-based applications have been developed for local area networks (LAN), such as Ethernet [33]. As these networks have become interconnected by store-and-forward packet switches in order to build extended LANs, extensions and new algorithms have been proposed to support the migration of the applications and provide efficient routing for multicast across these interconnected networks [32, 34]. Today, multicast routing has a number of practical applications, some of which are multimedia streaming, video conferencing, database management, distributed computation and real-time workgroup activities such as exchanging files, graphics or messages [31].

As a result of the developments in the field of wireless communications, multicast routing protocols also play an important role in mobile ad hoc networks to provide communication and coordination among a given set of nodes. It is particularly advantageous to facilitate multicast rather than use multiple unicast in ad hoc networks, where scarce resources are shared in the wireless medium. However, conventional wired network multicast protocols do not perform well in the ad hoc domain due to the unreliable nature of the wireless links and dynamic network topology. The multicast protocols used in the conventional wired networks usually require global knowledge on routing such as link state or distance vector structures, which are not feasible for ad hoc networks [11]. Therefore, several ad hoc multicast routing protocols are proposed to address the problem, some of which are categorized in Table 2.1 and summarized in the following sections.

Table 2.1. Chronology of ad hoc multicast routing protocols

| Multicast Protocol | First Publication | Multicast Topology | Initiation Type | Flooding Control | Periodic Control |
|---|---|---|---|---|---|
| AMRoute | 1998/08 | Shared-tree | Source | Yes | Yes |
| ASTM | 1998/11 | Shared-tree | Receiver | Yes | Yes |
| FGMP | 1998/11 | Mesh | Receiver | Yes | Yes |
| AMRIS | 1998/11 | Shared-tree | Source | Yes | Yes |
| CAMP | 1999/03 | Mesh | Receiver | No | No |
| MAODV | 1999/08 | Shared-tree | Receiver | Yes | Yes |
| MCEDAR | 1999/09 | Source-tree | Receiver | Yes | No |
| ODMRP | 1999/09 | Mesh | Source | Yes | Yes |
| BEMR | 1999/10 | Source-tree | Receiver | Yes | No |
| DDM | 2000/07 | Source-tree | Receiver | Yes | Yes |
| NSMP | 2000/08 | Mesh | Source | Yes | Yes |
| ABAM | 2000/09 | Source-tree | Source | Yes | No |
| MZR | 2001/07 | Source-tree | Source | Yes | Yes |
| MMA | 2001/08 | Shared-tree | Receiver | No | Yes |
| ITAMAR | 2001/10 | Source-tree | Source | n/a | n/a |
| DCMP | 2002/06 | Mesh | Source | Yes | Yes |
| LTM | 2002/10 | Source-tree | Source | n/a | n/a |
| WBM | 2002/11 | Source-tree | Receiver | Yes | No |
| RDG | 2003/03 | Mesh | Receiver | Yes | Yes |
| PLBM | 2003/05 | Source-tree | Receiver | No | Yes |
| PPMA | 2004/06 | Source-tree | Source | n/a | n/a |
| PUMA | 2004/10 | Mesh | Receiver | No | Yes |
| MANSI | 2005/02 | Shared-tree | Receiver | Yes | Yes |

### 2.1. Classification of Ad Hoc Multicast Routing Protocols

There are various categories into which ad hoc multicast routing protocols can be classified. For instance, they can be classified by how multicast connectivity is established and maintained. In a source-initiated approach, a multicast tree or mesh is constructed per sender, where the formation of the multicast group is initiated by the source. The source polls the network periodically with join request packets. Receivers wishing to join the multicast group respond with join reply packets when the propagated request reaches them. In a receiver-initiated approach, a single multicast connection is shared by all senders of the same group. A receiver floods a join request packet to search for a path to a multicast group. One common technique used with this approach is to assign a node, known as the rendezvous point or the core, to accept join requests from members. The multicast connection then consists of shortest paths from the core to each of the members.

Another possible classification is based on the operation type of the ad hoc multicast protocols. According to this classification, a protocol can be either proactive or reactive. Proactive protocols typically require table-driven preparation activities, whereas in reactive protocols the process is on-demand. This classification is inherited from ad hoc routing protocols. It is shown by previous research that, generally, on-demand approaches are better-suited than table-driven ones due to the dynamic nature of ad hoc networks.

Based on the multicast topology, multicast routing protocols are grouped into two types: tree-based and mesh-based. In tree-based protocols, there exists only one possible path between a source-destination pair, whereas in mesh-based protocols, there may be more. Tree-based protocols are further categorized into shared-tree and source-tree topologies. In the former, all members of a multicast group are connected via a single shared tree, whereas in the latter, a group consists of multiple trees rooted at their respective sources. While tree-based protocols are more efficient in terms of resource usage, mesh-based protocols are more robust to the changes in the network. In the following sections, previous research on ad hoc multicast routing is examined based on this topological classification. As explained in Section 3, AQM initially constructs a multicast tree and evolves it into a mesh during the session. Therefore, it is appropriate to investigate its predecessors with this perspective.

## 2.2. Shared-Tree-Based Multicast Routing Protocols

In a tree-based multicast routing protocol, a node accepts packets only when they come from another node which a tree branch has been established with. Thus, there is only a single path between a sender-receiver pair. A subset of the tree-based protocols constructs a single, shared multicast routing tree spanning all the group members. Such a tree may not be optimal for the individual sources, but they are more scalable when the number of sources in a session or the number of multicast sessions increases. Some of these protocols are presented in the following sections.

### 2.2.1. Ad Hoc Multicast Routing Protocol

Ad hoc multicast routing (AMRoute) is a shared-tree protocol running over an underlying mesh [35, 36]. It has two main components: mesh creation and tree creation. Dynamic logical group cores create and maintain the multicast tree and add new members. Neighbouring tree nodes are interconnected via unicast tunnels. It is the underlying unicast protocol's responsibility to maintain connectivity among members.

After a mesh is created, all cores periodically broadcast join request messages to discover mesh neighbours. When a core or non-core member receives this request, it replies with a join acknowledgement message. If a member does not wish to join the session or wishes to leave, it sends a negative acknowledgement to its neighbours. The cores also send periodic tree creation messages along unicast tunnels. Members receiving these non-duplicate messages forward them to other mesh links and mark incoming and outgoing links as tree links. If a link is not going to be used, a negative acknowledgement is sent back in response to the tree creation message along the incoming link. In the mesh merging process, multiple active cores may coexist. Nodes receive tree creation messages from different cores. A core resolution algorithm decides on a unique core for the mesh.

### 2.2.2. Adaptive Shared-Tree Multicast Routing

Adaptive shared-tree multicast (ASTM) protocol facilitates a rendezvous point as the root of the multicast tree to combine the advantages of per-source and shared trees [13, 37,

38]. Receivers send their join requests periodically to the rendezvous point for the creation of the multicast tree with a list of sources they expect to get data packets from. Sources send their data to the rendezvous point to be forwarded to the receivers. Intermediate nodes on the path between the source and the rendezvous point may not forward these packets to other nodes if the protocol is operating in the unicast sender mode. However, forwarding to other nodes known to be receivers of the source is allowed in the multicast sender mode of the protocol.

ASTM also allows sources to multicast without the intervention of the rendezvous point, if there are nodes which belong to the tree between the rendezvous point and themselves. This method is called adaptive per source multicast routing. According to this method, packets might travel paths other than the shortest path. Therefore, a receiver should not be forced to travel on the tree rooted at the rendezvous point if the source is nearby. Moreover, receivers should be allowed to switch between the rendezvous-point-rooted tree and the per-source tree dynamically using relative path length and link load as decision criteria. Switching is possible for a receiver by sending a join request to the source for a forwarding path and letting the record for the source-receiver pair expire in the forwarding list of the rendezvous point. When nodes move and the relative path length becomes infeasible to use that path, the receiver can switch back to the shared forwarding tree rooted at the rendezvous point.

### 2.2.3. Ad Hoc Multicast Routing Protocol Utilizing Increasing ID-Numbers

The ad hoc multicast routing protocol utilizing increasing ID-numbers (AMRIS) builds a shared tree for each multicast session [39]. A single source with the smallest member ID broadcasts the new session packet. Other nodes generate their own IDs based on the ID and hop count values they receive in this packet in an increasing manner, put the new ID to the packet and rebroadcast it. Multiple copies of the new session packet are not processed. The new session packet propagates outward from the session source in the form of an expanding ring. Eventually, all nodes have their own IDs regarding this session. They also keep a neighbour status table.

When a node sends a join request to its neighbouring potential parent, the latter checks the ID of the requester and replies with an acknowledgement if the ID is greater than its own. Otherwise, the parent sends back an error message and waits for the requester to increase its ID. If the parent itself is not yet a member of the session, it sends its own passive join request to its potential parent. The process is completed when the requesters, upon receiving acknowledgements from their parents, send confirmation messages to show that they have actually selected their parents for the session. A node wishing to join a session without a potential parent around broadcasts a request with a certain range. All nodes within that range must attempt to join the multicast tree with their own passive join requests. A node leaves a session by sending a session leave packet to its potential parent.

### 2.2.4. Multicast Ad Hoc On-Demand Distance Vector Routing

Multicast ad hoc on demand distance vector (MAODV) routing protocol tries to provide unicast, multicast and broadcast capability to its users [7, 40, 41]. It is derived from AODV [42, 43]. It is a tree protocol whereby routes are discovered on demand and use a broadcast discovery mechanism. Multicast and unicast routing information help each other to learn new routes. The multicast group leader maintains a group sequence number and broadcasts it periodically, which is very important to keep the routing information fresh.

A node wishing to join a multicast group generates a route request with its join flag set. If the multicast group leader is known, it can be found in the request table and the request is unicast to the leader. Otherwise, the request has to be broadcast. Only the leader or members of the multicast group with a higher sequence number than that in the join request may respond to the request by generating a route reply and unicasting it back to the requester. Other nodes may only rebroadcast or forward the packet. If the requester does not receive a route reply after a certain number of attempts, it becomes a group leader. Nodes receiving join requests update their route and multicast tables with the downstream next hop information. Nodes receiving reply messages update their tables with the upstream next hop information. They increment hop counts and forward the message to the node that has originated the request, which eventually receives several route replies, selects the best one in terms of highest sequence numbers and lowest hop count and

enables that route by unicasting a multicast activation message to its next hop neighbour on the selected path. Intermediate nodes receiving the activation message enable their multicast table entries for the requester. If they are already multicast group members, further propagation of the message is not necessary. Otherwise, they unicast it upstream along the best route according to the replies they received previously. Nodes having generated or forwarded replies, but not received any activation, delete their entries after a timeout. Figure 2.1 illustrates the three phases of such a join operation in MAODV.

The maintenance of the tree is accomplished by means of an expanding ring search started by the downstream node with a fresh join request, which contains the hop distance of the requester to the group leader and the last known sequence number. This request can be answered only by those tree members which are closer to the group leader and have a greater sequence number for the session. Nodes wishing to leave the group prune themselves by unicasting a multicast activation message to their next hop with the prune flag set.

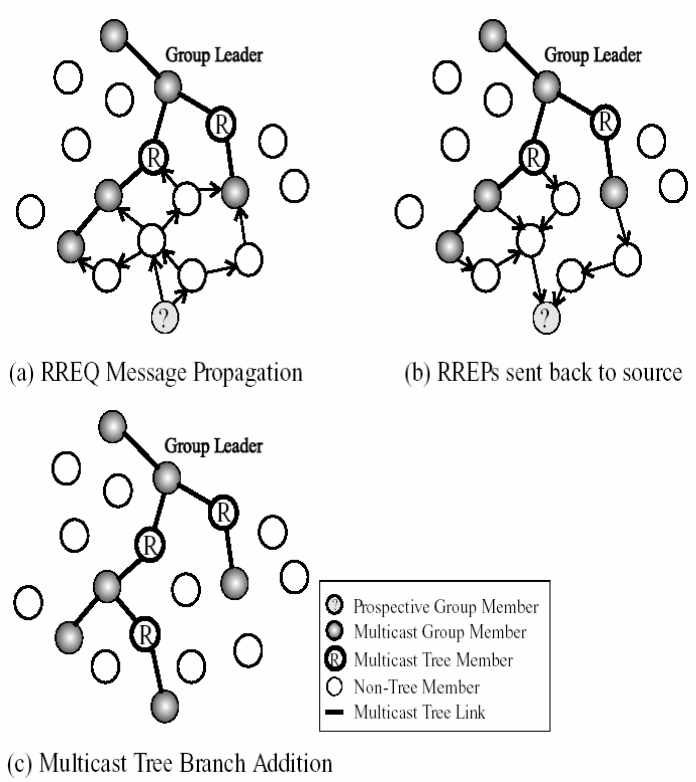



Figure 2.1. Multicast join operation of MAODV [40]

### 2.2.5. Mobile Multicast Agents

A multicast routing algorithm based on mobile multicast agents (MMA) in ad hoc networks uses a two-level hierarchy, where a special subset of network nodes form a spine to act as a virtual backbone on top of a clustered structure [44]. Spine nodes are called MMAs, which are responsible for multicast tree discovery and maintenance. MMAs are also used as relay hosts, so that the multicast tree is composed of a sender host, MMAs and multicast group members. Route information is only stored in MMAs. Thus, tree discovery is simplified and the time to find the tree is reduced.

According to this algorithm, the multicast group is only composed of MMAs that dominate some multicast members. Thus, only MMAs are flooded with route request packets and control overhead is reduced. A clusterhead is automatically selected as a MMA. Spine nodes function as MMAs of their dominating nodes. Spine construction is based on an approximation to a minimum connected dominating set. The MMA multicast algorithm is based on AODV [42, 43].

When a node wants to join a multicast group, it sends a request packet to its MMA, which checks its routing table for that multicast group. If MMA already knows the multicast tree valid for that group, it uses this tree. If no information is available, the MMA broadcasts a route request to the spine to initiate a route discovery process. Each MMA looks up its own routing table for the desired multicast group tree and sends back a reply if it does. Otherwise, it broadcasts the request further to its neighbouring MMAs. If an intermediate MMA has partial information on the desired group, then it adds this information to the request packet to show that routes to some members have been found and broadcasts the request further. It also appends its cached routes to a reply packet and sends it back to the requesting MMA.

Non-spine nodes are periodically queried by dominating MMAs to detect new entries or absence of old ones in the domination area. Route errors and acknowledgements are used for route maintenance. The load of the protocol is mainly on MMAs. Time must be spent to construct a spine by seeking MMAs before communication.

### 2.2.6. Multicast for Ad Hoc Networks with Swarm Intelligence

The multicast for ad hoc networks with swarm intelligence (MANSI) protocol is inspired from the nature, where especially social insects such as ants and honeybees work collectively as a group to achieve globally optimized behaviour, although each individual has limited intelligence [45]. Similarly, MANSI utilizes small control packets which deposit information at the nodes they visit. This information is used by other control packets later. MANSI adopts a core-based approach to establish connectivity among members. The core is the first sender of a multicast session. The initial announcement is flooded by the core towards the receivers and the nodes on the reverse paths serve as forwarding nodes to the group members. MANSI determines a forwarding set using the intermediate nodes shared among multicast senders. This set connects all group members for each multicast group and evolves in such a way that its cost decreases during the lifetime of the session. MANSI tries to reduce the number of nodes used to establish connectivity. For this purpose, nodes tend to choose paths that are partially shared by others to reduce the size of the forwarding set. Periodic exploration messages are deployed by members to search for new forwarding nodes with lower cost. Active forwarding members reply to these search packets. If the cost of the new path is lower for the intermediate and requesting nodes, the requester switches to the new route and the old one expires. Figure 2.2 shows a change of multicast routes to reduce the number of forwarding nodes in the tree and achieve lower cost.

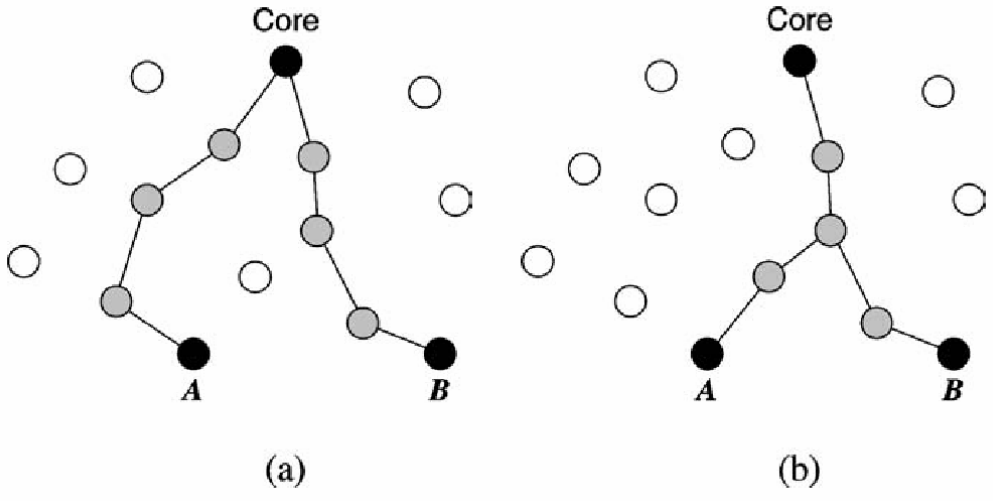



Figure 2.2. Multicast connectivity examples in MANSI [45]

## 2.3. Source-Tree-Based Multicast Routing Protocols

In contrast to the shared-tree approach, multicast groups generated by a source-tree-based protocol consist of multiple trees rooted at their respective sources. Source-tree-based protocols perform better than shared-tree-based protocols under heavy traffic since they achieve more efficient load balancing. Some of these protocols are presented in the following sections.

### 2.3.1. Multicast Core-Extraction Distributed Ad Hoc Routing

Multicast core-extraction distributed ad hoc routing (MCEDAR) is a protocol trying to bring mesh robustness and tree efficiency together [46]. It is an extension to CEDAR, which establishes a core network and uses it as a route management infrastructure [47, 48]. Each node selects a neighbour with maximum degree and maximum number of joined nodes as its core. When a node joins a core, it broadcasts its neighbourhood information. Thus, each core has enough local topology data to reach the domains of other cores. Core broadcast messages have the form of unicast between cores and are sent to discover destination locations and topology information. In MCEDAR, the multicast group is a subgraph extracted from the core.

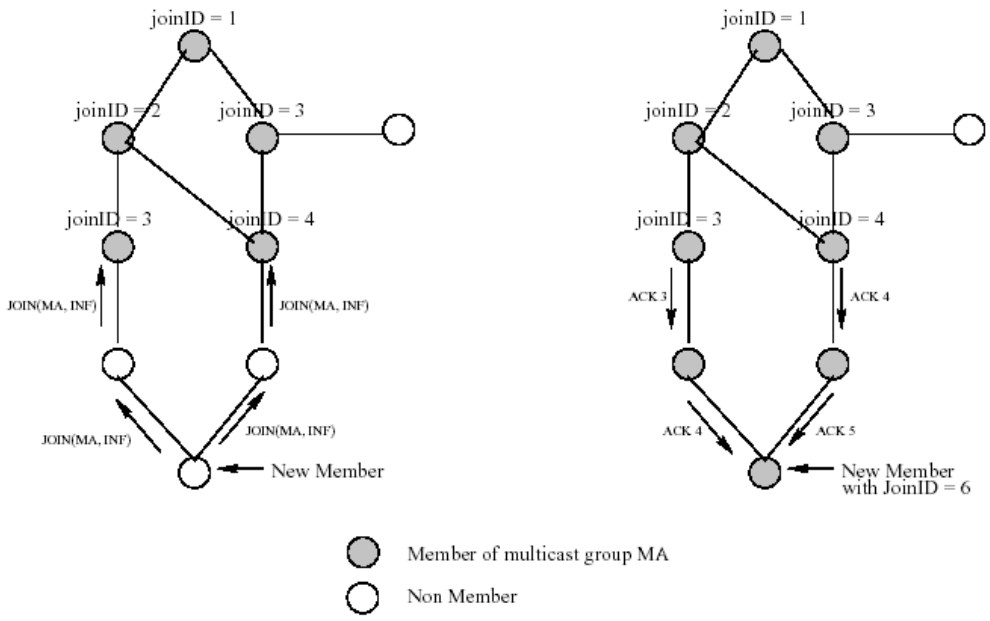



Figure 2.3. The join protocol of MCEDAR [46]

In order to join the multicast group, a node requests its dominating core to broadcast a join request with a join ID set to infinity. Non-member cores forward the request. A member core receiving the request replies with a join acknowledgement if its own join ID is smaller than that of the requester. It also forwards the request. Intermediate nodes accept a limited number of acknowledgement messages for a request. They become members by setting their join ID to the maximum of their current ID and the arriving ID. They forward the acknowledgements with their new join ID and add corresponding entries to their parent and child sets. If an intermediate node rejects an acknowledgement, it suppresses the message and notifies its parent to be removed from the child set. Eventually, the core which has sent the request receives the acknowledgements and the node joins the group. Figure 2.3 displays the join process of MCEDAR.

Every member tries to keep a certain number of parents in the multicast group by sending new join requests periodically. Rejoin requests are issued with the current ID when all parents are lost, which causes a global ordering on join IDs and can trigger cascade changes in the entire graph.

### 2.3.2. Bandwidth-Efficient Multicast Routing

Bandwidth-efficient multicast routing (BEMR) tries to achieve bandwidth efficiency by utilizing a small number of control packets and multicast efficiency by decreasing the number of transmissions for packet delivery in the multicasting process [49, 50]. For this purpose, newly joining nodes try to find the nearest forwarding multicast member and tree reconfiguration is done only when a link break is detected, avoiding the periodic transmission of control packets.

When a new node broadcasts a join request, each node receiving the request adds its ID and increments the hop count before flooding it back to the network. The hop count indicates the number of new nodes that need to be added to the multicast group in order to create a path from the group to the node originating the request. Forwarding nodes receive some of these requests, choose the best hop alternative and send a reply packet along the selected path. The requester eventually receives multiple replies, chooses the best hop alternative and sends a reserve packet along the same path. All nodes on this path become

forwarding nodes. The route setup process is illustrated in Figure 2.4. To leave a session, nodes send prune messages to their upstream neighbours. If the neighbour does not have any other downstream multicast member, it also leaves the session.

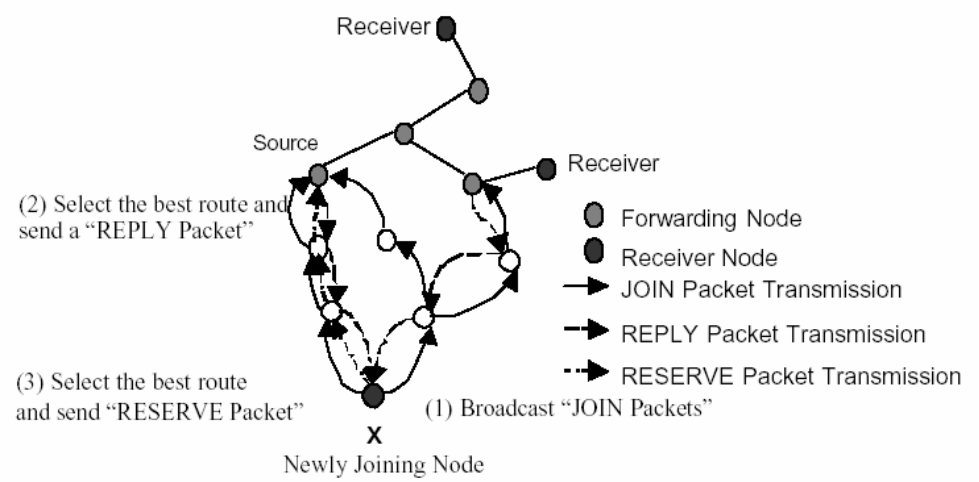



Figure 2.4. Route setup in BEMR [50]

The routes in BEMR can later be optimized by removing unnecessary forwarding nodes. The optimization process creates a shorter route when a forwarding node or receiver receives a multicast packet with a smaller hop count, as a result of moving into the range of an upstream forwarding node. In this case, the node sends a reserve packet to the new upstream node and a leave packet to the old one.

There are two schemes in BEMR to recover from link failures. In the first scheme, the upstream node detects the failure and looks for a new route to the lost downstream node by flooding a broadcast-multicast control packet locally. When the downstream node receives this packet, it sends a reserve packet back and rejoins the multicast group. In the second scheme, the downstream node tries to reconnect to the multicast group by flooding a join packet locally. When nodes from the multicast group receive the packet, they reply. The downstream node selects its new upstream node and sends a reserve packet to it.

### 2.3.3. Differential Destination Multicast Routing

Differential destination multicast (DDM) lets source nodes manage group membership as admission controllers and stores multicast forwarding state information

encoded in headers of data packets to achieve stateless multicast routing [51-53]. The protocol is intended for small multicast groups and is not general-purpose.

In DDM, join messages are unicast to the source, which tests admission requirements, adds the requester to its member list and acknowledges it as a receiver. The source needs to refresh its member list in order to purge stale members. It sets a poll flag in data packets and forces its active receivers to resend join messages in order to remain in the member list of the source. Leave messages are also unicast to the source, which removes the leaving member from its list. Forwarding computation is based on destinations encoded in the headers. During this process, a node has to check the header for any DDM block or poll flag intended for it and take the appropriate actions.

In order to achieve stateless multicast, DDM keeps the destination addresses of a multicast session encoded in the header of the data packets, which are sent to the next node using the underlying unicast protocol. However, there is a second operation mode in DDM, whereby nodes remember the destination addresses and there is no need to store them in data packets. Instead, the upstream node informs the downstream node with a data packet containing a differential type address block when the list of destinations changes.

### 2.3.4. Associativity-Based Ad Hoc Multicast Routing

Associativity-based ad hoc multicast (ABAM) builds a source-based multicast tree [54, 55]. Association stability, which is achieved when the number of beacons received consecutively from a neighbour reaches a threshold, helps the source select routes which will probably last longer and need fewer reconfigurations. The concept as well as the route selection algorithm is based on the Associativity-based routing (ABR) protocol [55, 56]. The tree formation is a process consisting of three phases. The first phase is initiated by the source by broadcasting a message to the group it intends to reach, whereby it specifically identifies its receivers. In the second phase, valid receivers, which already know possible routes to the source, run a route selection algorithm to select and reply with routes of highest association stability. Upon receiving the replies, the source runs a tree selection algorithm to find common links, builds the shared-link multicast tree and sends a setup message to its receivers, which is the third phase of the process.

To join a multicast tree, a node broadcasts a request, collects replies from group members, selects the best route and sends a confirmation. Tree reconfigurations occur when the associative property is violated due to node mobility. Broken links are repaired by the upstream members of the tree through the use of local queries. If the repair attempts of the upstream nodes fail, the disconnected receiver sends a new join query packet. To leave a multicast tree, a notification is propagated upstream along the tree until a branching or receiving node is reached.

### 2.3.5. Multicast Zone Routing

The multicast zone routing (MZR) protocol tries to limit the flooding of the control packets generated by a node during the member searching process within its zone [57]. The concept is borrowed from the zone routing protocol (ZRP), which associates each node with a routing zone [58]. Routing inside a zone is table-driven, whereas routing across zones is on-demand. In its multicast version, the source first builds a tree inside its zone by unicasting a tree create packet to the nodes within the zone. Interested receivers reply with acknowledgement packets. Then, the tree is extended outside the zone by the source with a tree propagation packet sent to the border nodes of the zone. The border nodes convert these packets into tree create packets and distribute them within their zones. Once the multicast tree is created, the source periodically transmits tree refresh packets. If any node on the tree does not receive these packets for a certain period of time, it removes the tree from its records. Downstream nodes have to detect link breaks and rejoin the group. Similar to the tree generation process, the join packet is first sent to the zone nodes. If no join acknowledgement is received, a join propagate packet is sent to the border nodes.

### 2.3.6. Weight-Based Multicast

The weight-based multicast (WBM) protocol provides a receiver with the options of joining the multicast session at the nearest point to either the tree or the source [11, 59]. The protocol facilitates a weight when deciding on the entry point. Thus, it considers both the number of intermediate nodes which need to be added to the tree, as well as the hop distance to the source before joining a multicast group.

A new receiver about to join a group broadcasts a join request, copies of which are forwarded until they reach some nodes on the tree. Several replies are returned, which contain the hop distance of the replying node both to the source and to the requesting node. The new receiver selects the best alternative by calculating a weighted sum of these values and sends a join confirmation along the selected path. The join weight parameter depends on the network load and the size of the multicast group.

WBM uses a localized prediction technique for the maintenance of the multicast tree. Each node maintains a neighbour multicast tree table to keep the hop distance information of the tree nodes. When a downstream node receives data from its parent node, it predicts the time duration for which the nodes remain within the transmission range of each other. If this duration is below a threshold, then the node decides that it needs a new parent, raises a handoff flag in the data packet and forwards it. A neighbouring node overhearing such a packet looks up its neighbour multicast tree table and sends a handoff packet back to the node requiring handoff if there is a tree member node which can be the new parent. Upon receiving several handoff packets, the node requiring handoff selects one of them and send a handoff confirmation to the neighbour, which forwards it to the tree member so that the node requiring handoff can rejoin the multicast tree.

### 2.3.7. Preferred-Link-Based Multicast

The preferred-link-based multicast (PLBM) protocol uses the notion of preferred links and facilitates the two-hop local topology information for efficient multicast routing [11, 60]. Each node maintains a list of its two-hop neighbours, which is kept up-to-date by means of small control packets called beacons and transmitted periodically by every node. Each node also maintains information on the multicast tree.

A node wishing to join the multicast tree checks its list of neighbours to see whether the multicast source, a multicast member or a forwarding node is around. If this is the case, it sends a join confirm packet to this neighbour. Otherwise, it checks the same list to see whether there are neighbours which can be preferred to send a join query. The decision on the eligibility of the neighbours is made by an algorithm originally developed for the PLBR protocol [11, 61], which is the unicast ancestor of PLBM. According to one

implementation of this algorithm, preference is given to those neighbours with a higher neighbour degree. As higher degree neighbours can reach more nodes, a few of them is sufficient to cover all the nodes in the two-hop neighbourhood, which reduces the number of broadcasts. A subset of the eligible nodes are inserted into the preferred list field of the join query, which is then sent away to be further forwarded by those nodes.

Upon receiving the join request, a node on the preferred list sends a join reply if it is already connected to the multicast tree. Otherwise, it forwards the packet after generating its own preferred list. When the originator of the join request receives a reply, it sends a join confirm packet to the node which has sent the reply. Intermediate nodes receiving the join confirm packet mark themselves as connected and forward the packet. They also store the information on the next two-hop both upstream and downstream for this connection.

Figure 2.5 illustrates the propagation of the join queries of two ordinary nodes towards the multicast source, the join replies sent in response to the second query by two forwarding nodes and a join confirm packet sent by a node which discovers a forwarding node within its two-hop neighbourhood.

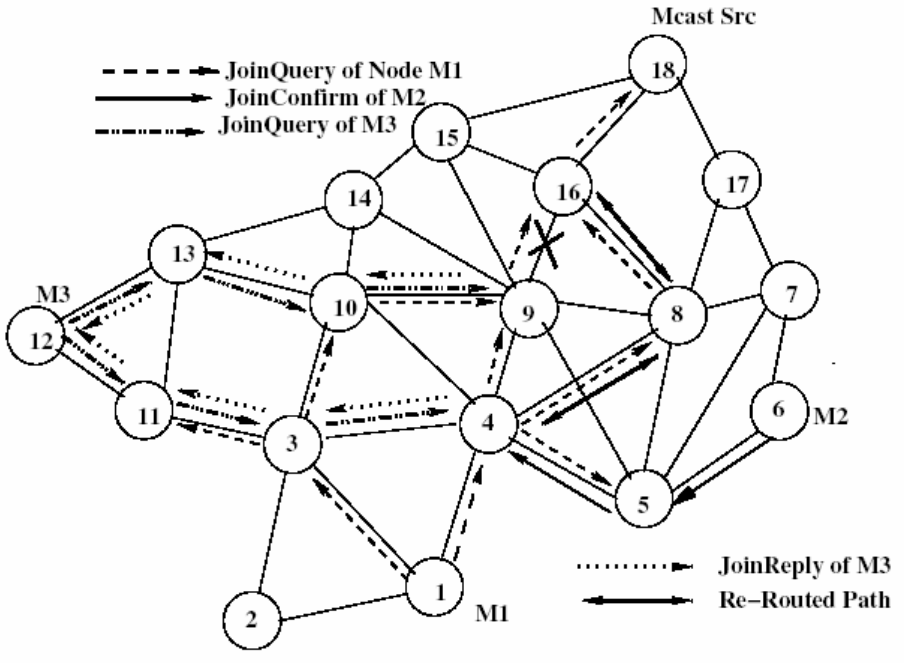



Figure 2.5. Examples for the join process of various types of nodes in PLBM [60]

Disconnected paths in PLBM are repaired by using the list of two-hop neighbours and the two-hop connection information of the tree. A node with downstream multicast members tries to connect to any tree node in its two-hop neighbourhood. If there are no such nodes, it initiates a join query and also sends another message to its downstream nodes to prevent them from sending their own join queries and keep the subtree connected.

## 2.4. Mesh-Based Multicast Routing Protocols

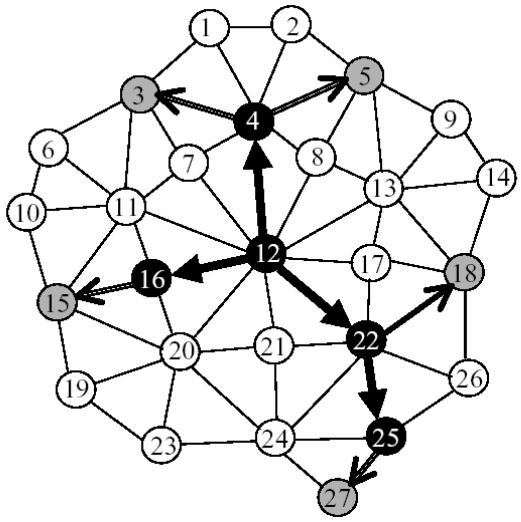
Since there is only a single path between senders and receivers in tree-based multicast routing protocols, they are vulnerable to the dynamics of the ad hoc network such as node mobility and link breaks. In contrast, mesh-based multicast protocols maintain a mesh consisting of a connected component of the network containing all the receivers of a group. They aim to construct a mesh that allows data packets to be transmitted over more than one path from a sender to a receiver to increase robustness at the price of redundancy in data transmission. Some of these protocols are presented in the following sections.

### 2.4.1. Forwarding Group Multicast Protocol

The forwarding group multicast protocol (FGMP) introduces the forwarding group concept [13, 62], which is later adopted by ODMRP. The main difference between FGMP and ODMRP is that the latter is a source-initiated protocol. FGMP keeps track of the nodes participating in packet forwarding, which are called the forwarding group. It uses this group, which is periodically refreshed, to limit the region of flooding. It uses flags instead of upstream or downstream link status information and makes use of the inherent broadcast capability of the wireless medium by exploiting the fact that in such an environment it is sufficient if a node just knows whether it has to forward data packets or not. Its multicast forwarding activities are based on nodes rather than links.

FGMP is a hybrid protocol between flooding and shortest tree multicast routing. The maintenance of its forwarding group can apply two schemes. In the receiver advertising scheme, join requests are issued by receivers periodically, which flood the global network. When these arrive at the server, it updates its member table with the new members, creates

a forwarding table from existing routing tables and finally broadcasts the forwarding table. Only those nodes that are among the next hop neighbours in the forwarding table take the packet into consideration. They set their forwarding flags, create their own forwarding tables and broadcast them in a similar manner until all receivers are reached. Figure 2.6 illustrates an example for this process. The sender advertising scheme works similarly but in the opposite direction, i.e., receivers periodically broadcast joining tables, which are forwarded upstream towards the sender. It is noteworthy that forwarding tables are not stored at the nodes. They are just temporarily created and broadcast.




Forwarding Table of Node 12:

| Receivers | Next hop |
|---|---|
| 3 | 4 |
| 5 | 4 |
| 15 | 16 |
| 18 | 22 |
| 27 | 22 |

(a)

Sender = {12}
Receivers = {3,5,15,18,27}
FG = {4,12,16,22,25}

Forwarding Tables flow
Mcast data flow

Forwarding Table of Node 22:

| Receivers | Next hop |
|---|---|
| 27 | 25 |

(b)

Figure 2.6. Forwarding group and tables in FGMP [62]

### 2.4.2. Core-Assisted Mesh Protocol

The core-assisted mesh protocol (CAMP) generalizes multicast routing trees into graphs by creating a shared mesh structure for each multicast group [63-66].Within a group, cores are used to limit the control traffic caused by join requests. Each node defines its predecessor in the multicast mesh from which it receives data as its anchor. In other words, anchors are those nodes that are expected to rebroadcast the multicast data they

receive to the routers downstream. Each node maintains a set of tables for routing, core-to-group mapping as well as anchor and multicast group management. When a node updates its anchor or multicast table, it sends a reporting message to all its neighbours.

The basic join mechanism is initiated by a host asking its router to join a group. CAMP assumes the availability of routing information from a unicast protocol. Thus, the router checks if there are any data-forwarding members of that group among its neighbours. If this is the case, the router directly announces its membership. Otherwise, it broadcasts a join request, which contains the information on the intended relay node towards the group core. Any member router of the intended multicast group can send a join acknowledgement. The requesting router and its relays become part of the group when they receive the first acknowledgement. Relays forward these replies towards their requester. A router leaves a multicast group if it has no member hosts and is not required as an anchor for any neighbouring node for that group. According to this mechanism, routers have to be group members to forward data packets of their hosts. However, CAMP has a secondary join mechanism which allows non-member sources to forward data in one direction only. Figure 2.7 illustrates the traffic flow in each of these schemes, whereby the solid arrows represent the flow of real data and the dashed arrows indicate overhead.

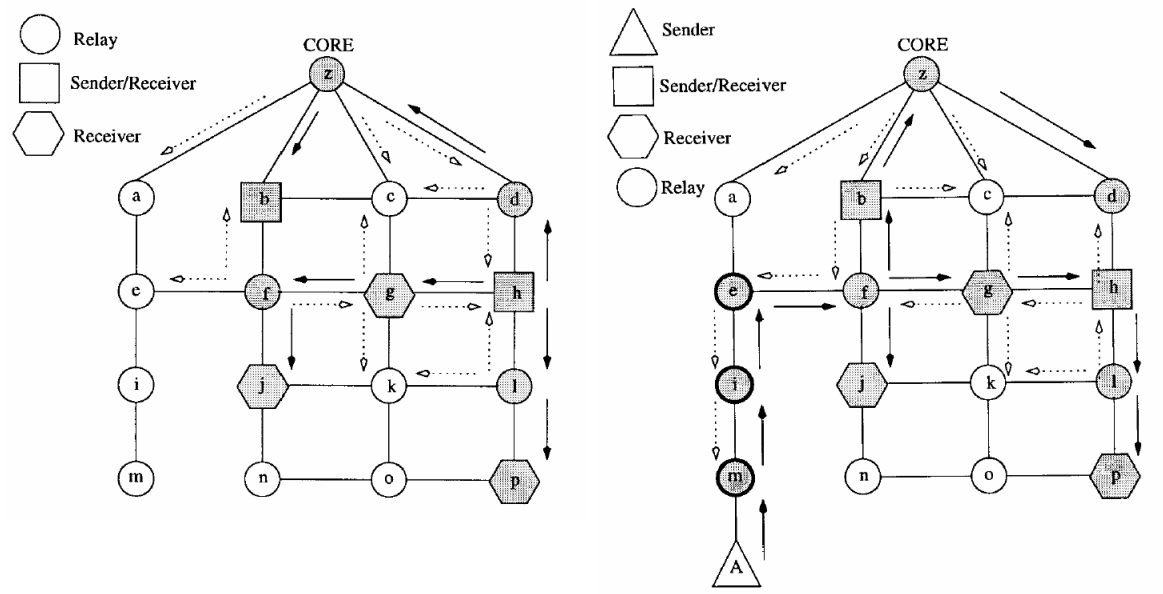



Figure 2.7. Traffic flow from router *h* and non-member source *A* in CAMP [65]

CAMP uses a scheme based on the transmission of heartbeat messages to ensure that the mesh contains all the reverse shortest paths. When the number of packets a mesh member receives from a multicast source via the reverse shortest path is under a threshold, the mesh member sends a heartbeat message along that shortest path towards the source. A router receiving a heartbeat forwards it if its successor towards the source is already a mesh member. Otherwise, it sends a push join and waits for an acknowledgement. A router receiving a push join sends an acknowledgement if it is the intended relay, is already a group member and has a path to the target of the push join. Then it forwards it to the next relay towards that target. Following this scheme, CAMP guarantees that every receiver in a multicast group knows a reverse shortest path to each source of that group.

### 2.4.3. On-Demand Multicast Routing Protocol

The on-demand multicast routing protocol (ODMRP) uses the concept of a forwarding group [6, 67-70], which is illustrated in Figure 2.8. Sources periodically broadcast join query messages to invite new members and refresh existing membership information as well as the routes between themselves and their receivers. When a node receives a non-duplicate join query, it stores the upstream node address in its routing table. If the time-to-live (TTL) field of the packet is greater than zero, the node also updates the join request with its own address, decrements the TTL field and rebroadcasts the packet. When a node decides to join a session, it sends a join reply packet along the reverse path to the source. When a node receives a join reply, it checks the table of next nodes to see if it is on the path to the source. If this is the case, it sets its forwarding group flag and broadcasts its own join reply after updating the table of next nodes. If a node receives a join reply to an already known source, however, no new join reply is sent.

Periodic join requests initiated by the source must be answered by session members in the form of join replies to remain in the group. Forwarding group nodes reset their flags if they do not receive any join replies. Each node maintains a routing table, a forwarding group table and a message cache to detect duplicate packets. Redundant routes in the mesh provide alternate routes for data delivery in the face of mobility and link breaks. Data packets can still reach destinations while the primary route is being reconstructed.

An extension is applied to ODMRP in order to achieve reliable data delivery [71]. According to this extension, sequence numbers are maintained by each node for data packets sent and received. Data packets are forwarded to each downstream group member in a round robin fashion, such that the sequence numbers are compared at both ends and missing packets are retransmitted by the sender. During the process, other neighbours receive data packets and update their sequence numbers as well. The process continues until the current node receives all the data packets it has missed and then is repeated with the next downstream group member.

In order to improve the local route recovery process of ODMRP without incurring a high maintenance overhead, another extension is proposed which introduces a passive data acknowledgement scheme, where control information is collected from data packets instead of beacon signals [72]. According to this extension, information on new routes is also collected from data packets, which enables dynamic local route maintenance. Since each node knows the status of its downstream forwarders, not too much route information needs to be collected in case of local route maintenance. The recovery process is performed by the upstream forwarding node to look for downstream nodes.

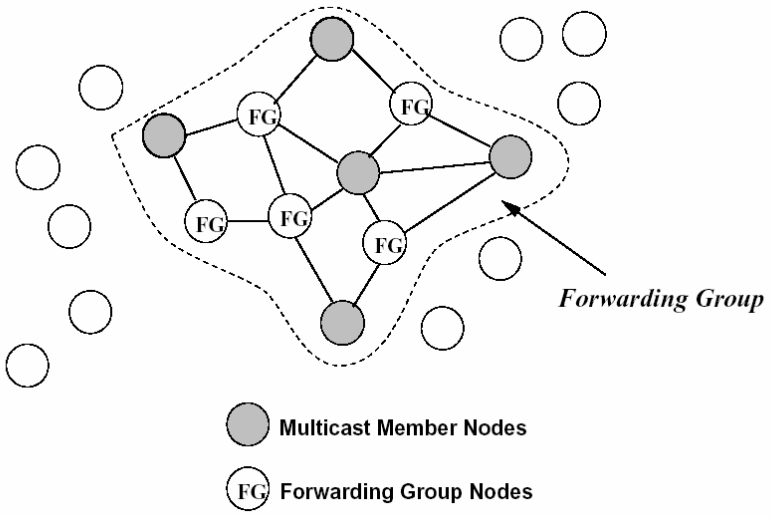



Figure 2.8. The forwarding group concept of ODMRP [70]

### 2.4.4. Neighbour-Supporting Ad Hoc Multicast Routing Protocol

Neighbour-supporting multicast protocol (NSMP) utilizes node locality to reduce route maintenance overhead [73, 74]. A mesh is created by a new source, which broadcasts a flooding request. Intermediate nodes cache the upstream node information contained in the request and forward the packet after updating this field. When the request arrives at receivers, they send replies to their upstream nodes. On the return path, intermediate nodes make an entry to their routing tables and forward the reply upstream towards the source.

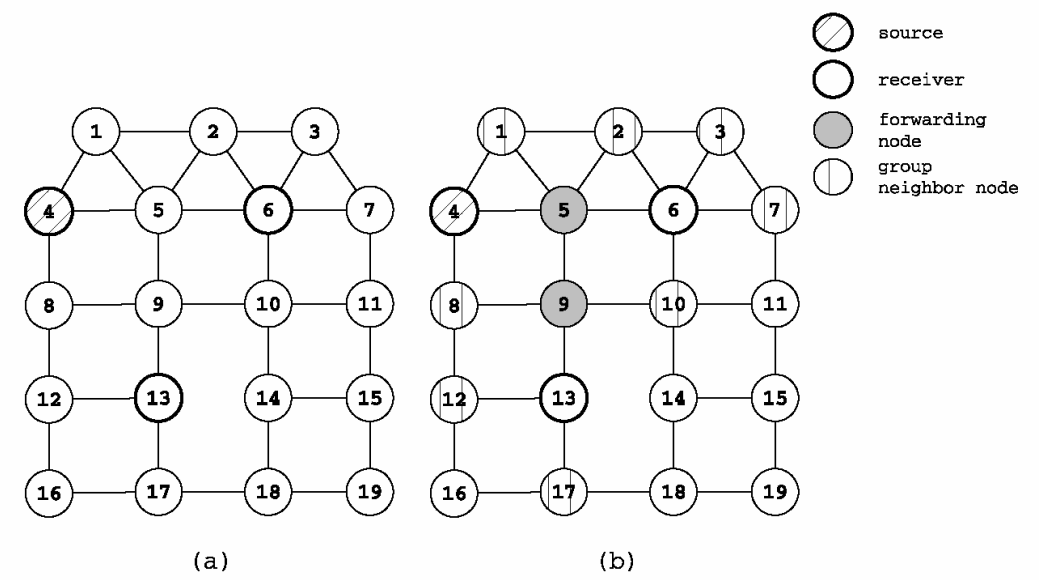



Figure 2.9. Multicast mesh creation in NSMP [74]

The mesh creation is illustrated in Figure 2.9. In order to maintain the connectivity of the mesh, the source employs local route discoveries by periodically sending local requests, which are only relayed to mesh nodes and their immediate neighbours to limit flooding while keeping the most useful nodes informed. If a new receiver wishes to join the multicast group, it waits for one of these local requests and then joins the group by sending a reply. Replies are sent back to the source to repair possible broken links. Nodes more than two hops away from the source cannot join the mesh with local requests. They have to flood member requests. On the other hand, sources periodically flood special request packets to recover the network from partitions. NSMP favours paths with a larger number of existing forwarding nodes to reduce the total number of multicast packet transmissions.

### 2.4.5. Dynamic Core-Based Multicast Routing Protocol

The dynamic core-based multicast routing protocol (DCMP) tries to reduce the control overhead by using a smaller number of forwarding nodes on a mesh and to prevent the packet delivery ratio from decreasing at the same time [75]. Its mesh creation protocol resembles that of ODMRP. However, DCMP defines active and core active sources, which are the nodes allowed to flood join request packets in the network and create a mesh.

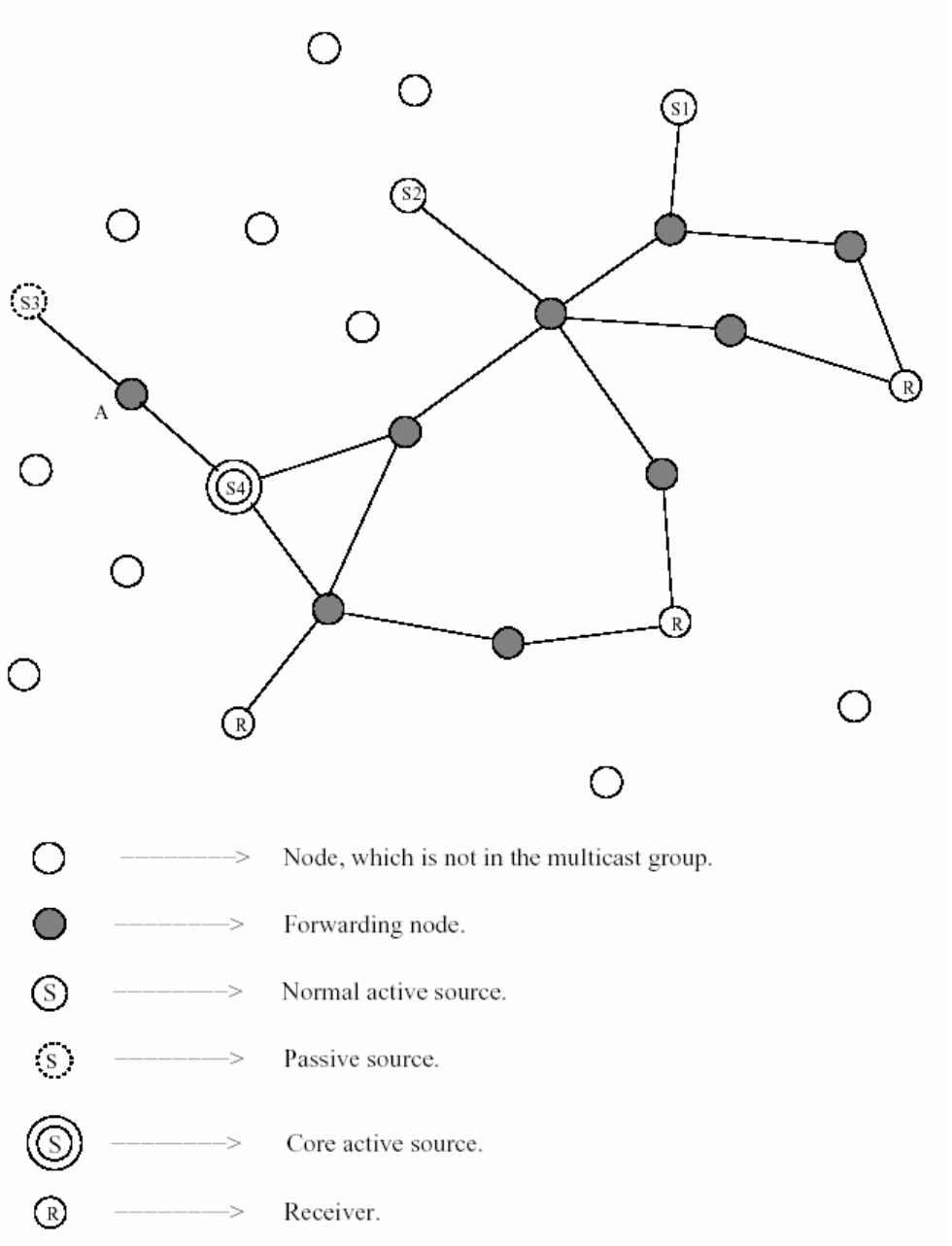



Figure 2.10. The mesh topology of DCMP [75]

Figure 2.10 illustrates a sample multicast mesh with various types of nodes. In addition to flooding join requests, active and core active sources can also send data packets. Passive sources, on the other hand, are not allowed to flood join request packets and can only send data packets to their associated active sources to be forwarded over the mesh. Thus, both the number of forwarding nodes and control overhead are reduced. The maximum number of passive sources is limited to guarantee that the mesh still has enough forwarding nodes. The maximum hop distance between a passive source and its active source is also limited so that the packet delivery ratio is not at stake.

### 2.4.6. Route-Driven Gossip

The route-driven gossip (RDG) protocol is developed to achieve probabilistic reliable multicast in ad hoc networks [76, 77]. According to the probabilistic reliability principle, if some group member sends out a flow of data packets, a certain group member receives a fraction of these packets with a certain probability. RDG nodes gossip uniformly about multicast data, acknowledgements and membership, also considering network parameters such as the availability of routing information. RDG can be deployed on any basic on-demand routing protocol, which has to be extended with control packets for group requests and replies.

In RDG, each node has a list of accessible group members, which is known as its view of the network. A node joins a multicast group by flooding the network with a request, whereby it announces its existence and searches for other members. All nodes receiving the request update their views and send replies. The requester also updates its view when it receives a reply. There is a periodic gossip task, whereby gossip messages are sent to a subset of randomly selected neighbours from a node's current view. Gossip messages are used to update receivers' views. The sender of a gossip is added to or removed from the views of the receivers of the gossip according to the current or intended membership status in the message. Gossip messages also carry data stored in the buffers of the nodes. A data packet is removed from the buffer after having been gossiped a predefined number of times. RDG yields reliable delivery results close to flooding with far less load on the network. Figure 2.11 illustrates the gossip process of RDG, where the number of randomly selected neighbours is two.

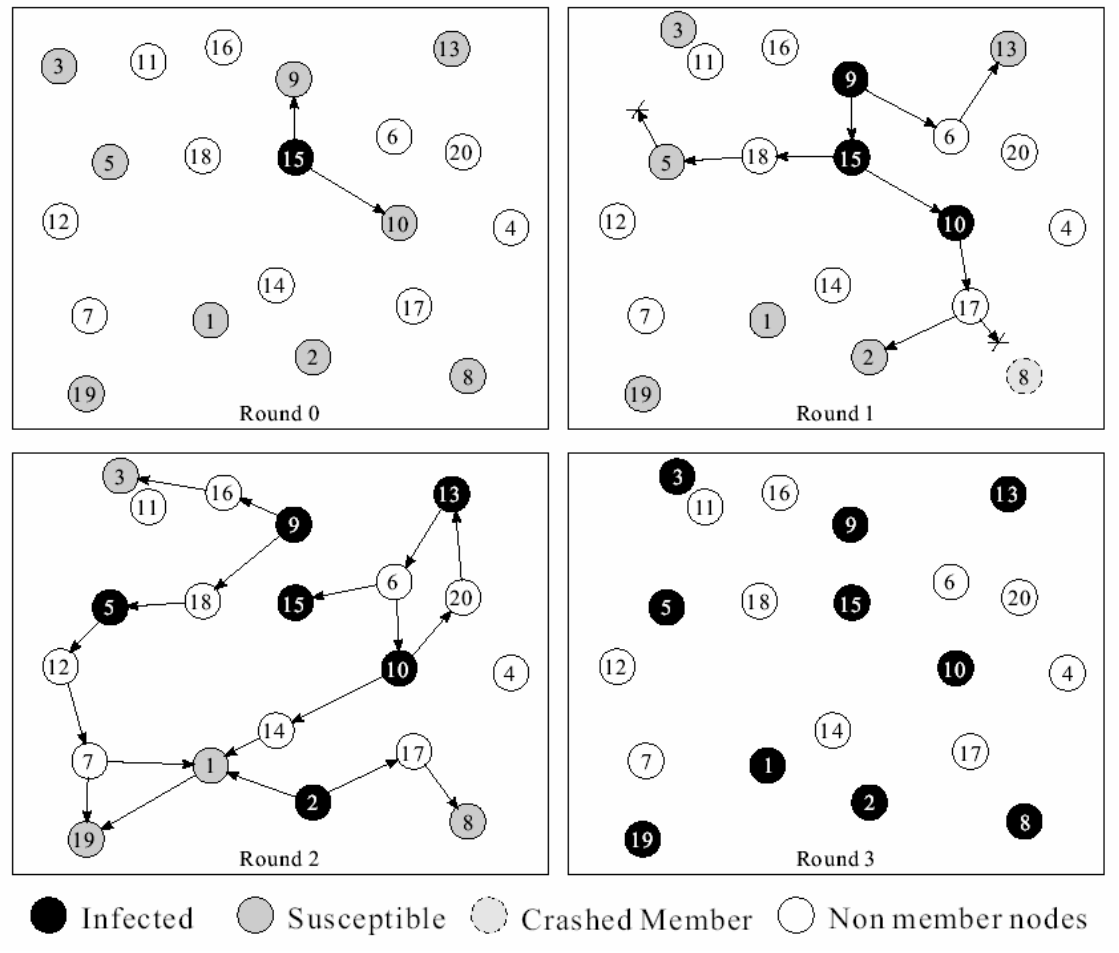



Figure 2.11. An example run of RDG [76]

### 2.4.7. Protocol for Unified Multicasting Through Announcements

The protocol for unified multicasting through announcements (PUMA) aims efficient and robust multicast routing in mobile ad hoc networks [78]. PUMA creates a shared mesh for each multicast group and has a dynamic core election mechanism where the first receiver of the group becomes the core. Subsequent receivers join the group with the help of the core. The core sends periodic multicast announcements. Connectivity lists are established at the other nodes using these announcements, such that each node is informed of at least one next-hop node towards the core. The connectivity lists are used for mesh creation and data routing. Each receiver connects to the core along all shortest paths and all intermediate nodes collectively form the mesh. Nodes prepare their own announcements based on the best announcement they receive, which is determined by its freshness and its sender's distance to the core.

When a node wishes to send data to a group, it forwards the data packets to the node from which it has received the best multicast announcement. If the connection to that node is broken, the sender tries the next best alternative. Initially, only receivers are mesh members. Non-receivers become mesh members only if they have mesh children according to their connectivity list. Eventually, all shortest paths from the receiver to the core are included to the mesh. An announcement is produced by a node if the core changes, the node receives a fresh announcement or the mesh member status of the node itself changes. During the data forwarding process, a non-member delivers its packets to its neighbour having sent the best announcement. Outside the mesh, only parents of receivers forward these data packets towards the core. As soon as the data packet enters the mesh, it is flooded by the mesh members without consulting the connectivity list. Figure 2.12 shows the mesh creation process of PUMA.

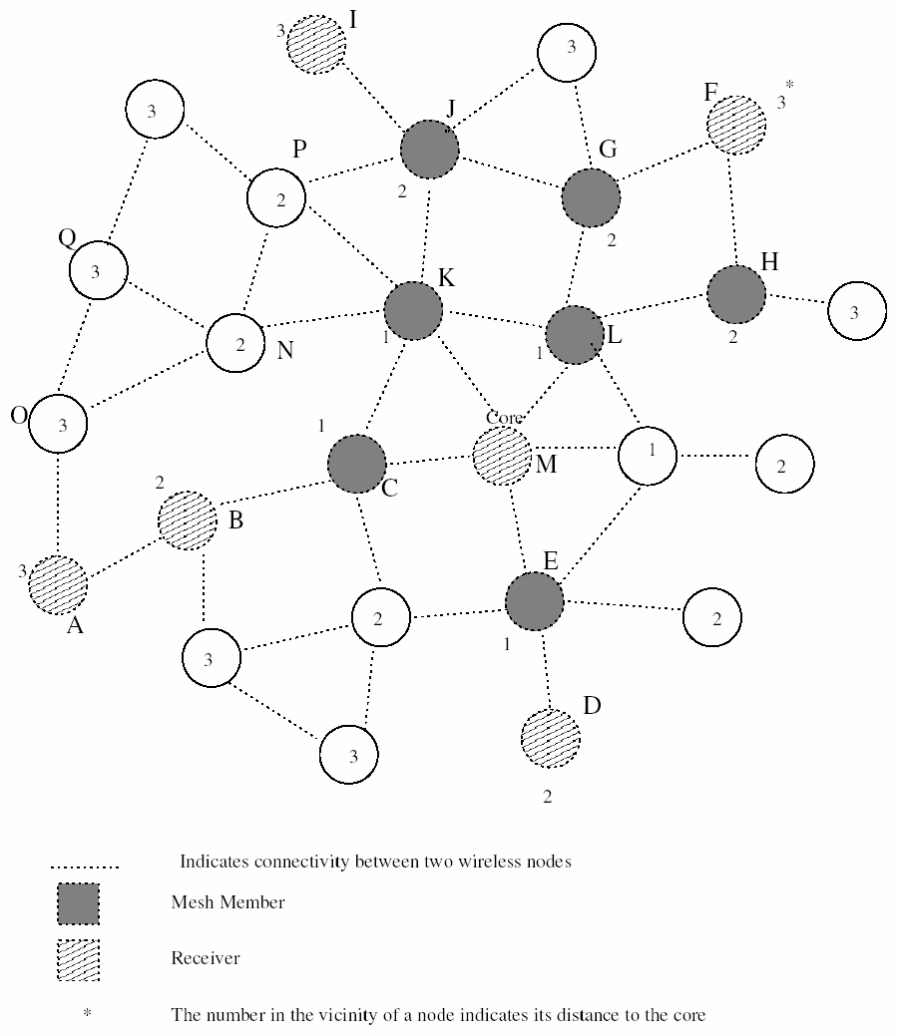



Figure 2.12. Mesh creation in PUMA [78]

## 2.5. Multicast Optimization Protocols

Some of the multicast protocols do not intend to solve the routing problem directly. They are rather interested in the development of a scheme which tries to optimize a certain aspect of the multicast network, such as node reliability, tree lifetime or route stability.

### 2.5.1. Independent-Tree Ad Hoc Multicast Routing

Independent-tree ad hoc multicast routing (ITAMAR) provides heuristics to generate a set of independent multicast trees, such that a tree is used until it fails and then replaced by an alternate tree [79, 80]. Maximally independent trees are computed by minimizing the number of edges and nodes which are common to the trees in question under the assumption that node movements are independent of each other. Some overlapping is allowed since totally independent trees might be less efficient and contain more links. Thus, the correlation between the failure times of the trees is minimal, which leads to improved mean times between route discoveries. New trees are computed when the probability of failure for the current set of trees rises above a threshold. Given a mobility pattern, it is important to estimate the time this happens. Then, instead of replacing a tree even if one link fails, an independent path algorithm can find a set of backup paths to replace the damaged part of the tree.

### 2.5.2. Lantern-Tree-Based Multicast

The lantern-tree-based multicast (LTM) protocol is a resource management scheme which can serve as the bandwidth reservation module of an on-demand multicast routing protocol [81, 82]. It provides end-to-end calculation and allocation of bandwidth from a source to a group of destinations by means of multipath routing. The scheme provides a single path if bandwidth is sufficient or a lantern-path if it is not. A lantern is defined as one or more subpaths with a total bandwidth between a pair of two-hop neighbouring nodes. A lantern path is a path with one or more lanterns between a source and a destination. Finally, a lantern tree is defined as a multicast tree which contains at least one lantern-path between any of its source-destination pairs. Figure 2.13 illustrates the lantern tree concept.

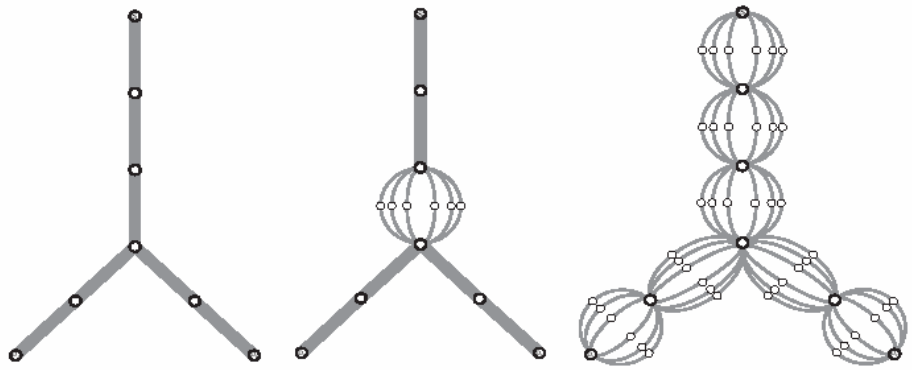


Figure 2.13. A tree, a lantern-tree and a worst-case lantern-tree in LTM [82]

According to LTM, the source sends a lantern-path request. The path is created if such a lantern exists. The process is repeated until a possible lantern-path arrives at the destination. Then a lantern-path is constructed. The replying paths from the destination back to the source are merged together to construct the lantern tree. The advantages of the lantern-tree approach are task sharing and higher stability. Sub-paths are responsible for a portion of the total bandwidth requirement and also for a fewer number of bandwidth requirements.

### 2.5.3. Probabilistic Predictive Multicast Algorithm

Probabilistic predictive multicast algorithm (PPMA) tracks relative node movements and statistically estimates their relative positions in the future to maximize the multicast tree lifetime by exploiting more stable links [83, 84]. Thus, by keeping track of the network state evolution, it tries to solve the drawbacks of lack of tree robustness and reliability in highly mobile environments. PPMA defines a probabilistic link cost as a function of energy, distance and node lifetime. It tries to keep all the nodes alive as long as possible. The protocol models the residual energy available for communication for each node, which is proportional to the probability of being chosen to a multicast tree. Nodes of low energy cannot join any more multicast trees.

The PPMA algorithm has a centralized and a distributed version. In the centralized version, a node has a set of potential fathers for a given number of hops. Higher priority is given to those nodes within transmission range having other children in order to exploit the

broadcast property of the wireless medium. The closest one among the potential fathers is chosen for power efficiency reasons. In the distributed version, a private cost is defined to find the minimum cost path to the source, in addition to a public cost to enable a node to join a tree. A new receiver finds the best public-cost path and joins the tree, whereas an old receiver changes its path if it finds a smaller private cost. The cost can be typically an entity such as energy consumption.

# 3. AD HOC QUALITY OF SERVICE MULTICAST ROUTING

The motivation behind QoS support for multicast routing in mobile ad hoc networks is the fact that multimedia applications are becoming increasingly important for wireless group communication. For an efficient ad hoc QoS multicast routing strategy, the implementation of QoS classes, negotiations between the network and its users, bounded end-to-end delay, jitter, as well as packet loss probability, bandwidth reservation and mobility management are very important. In the following sections, the structural components of AQM are presented, which address these issues.

AQM tracks the availability of resources within each node's neighbourhood based on current reservations made by that node for ongoing sessions and the requirements reported to that node by its neighbours. The allowed maximum hop count of the session is also taken into account in order to satisfy the delay requirements of the multimedia applications. The QoS status of the network is announced along with the QoS requirements of the session at the time of session initiation and updated periodically to the extent of QoS provision. In other words, session announcements are forwarded by a node only if the available QoS conditions allow the node to support that session in case a downstream node requests service later. Thus, nodes are prevented from applying for membership if there is no feasible QoS path for the session at the time of their request. When a node requests to join a session, a three-phase process consisting of request, reply and reserve steps is utilized whereby the updated QoS information helps the routers select one of the appropriate paths which can meet the service requirements of that session.

According to the protocol architecture, the algorithmic structure of an AQM node consists of four modules, which are the application module, the session module, the membership module and the network module. This modular architecture complies with the three layers in the network protocol stack generally concerned with multicast routing in mobile ad hoc networks [11]. The application module utilizes the services of the session and membership modules to satisfy the multicast requirements of the applications. The session module manages the distribution and maintenance of information on existing as well as new multicast sessions. The membership module is responsible for forming and

maintaining the multicast groups. Finally, the network module maintains the connectivity of the system and adapts it to the topological changes in the neighbourhood of a node. The architectural framework of AQM is illustrated in Figure 3.1, where the modules are placed with respect to the relations between them.

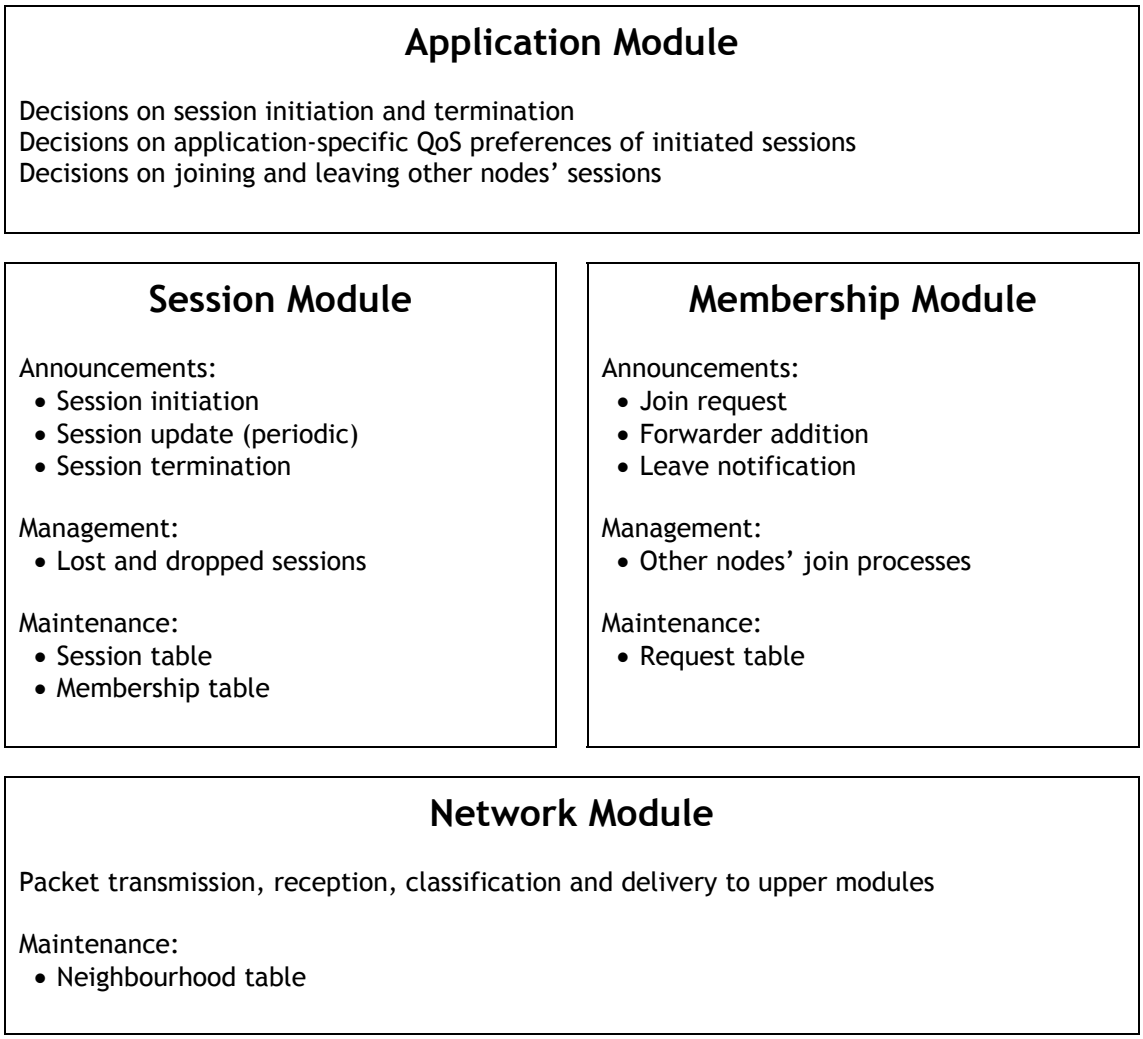



Figure 3.1. The architecture of the AQM protocol

### 3.1. Definitions

The AQM protocol makes use of various information entities such as packets and tables to maintain the multicast routing information. In addition, AQM tracks the state of a node within each session based on the task that the node accomplishes for that session. Finally, AQM runs several procedures to fulfil its multicast routing function based on QoS requirements by utilizing these packets, tables and the state information of the nodes.

Each AQM node tracks the state of other nodes as well as of its own as accurately as possible separately for each session. When a node has to take any action, such as handling a session initiation, update, termination or loss packet received from another node, it makes some decisions, which include whether to forward the packet, to destroy it, or to generate and send a new packet of a different type. Similarly, in case of a session join request, intermediate nodes have to decide whether to get involved when they receive any packets during the process and what the appropriate action is. All these decisions can be made by the receiver of the packet in an intelligent manner only after considering the state information they have on themselves as well as the packet senders. These decisions may also require changes in the current state information of a node. Regarding one session, there are six different states that a node can be in, which are given in Table 3.1.

Each AQM node maintains four different routing tables in order to take part in the management of the sessions and of the membership procedures. Information on existing sessions, node states regarding each session, join requests being processed and node connectivity is kept in these tables. The table of sessions and the table of members are populated by session initiation and update packets and the entries are deleted on the reception of loss and termination packets. The table of members is also populated by the join reply packets since they are obviously sent by session members. The table of requests is a temporary structure which is used by intermediate nodes during the request, reply and reserve phases of ongoing join requests. The table of neighbours keeps a node informed of the existence and resource allocation of all the nodes within its transmission range. The data in this table comes from the greeting messages of the neighbours and is deleted when a neighbour ceases to send such messages for a predefined period of time. The routing tables and their short descriptions are presented in Table 3.2.

Each AQM node uses ten different control messages to communicate with others in order to initiate or terminate sessions, join or leave them, take part in their session and membership processes and keep each other informed of the current status of a session as well as on their own existence. The control messages are defined in Table 3.3.

The structures of these information entities are presented in the following sections, where their usage within the AQM procedures is also explained in greater detail.

Table 3.1. Definitions for the node states in AQM

| Node State | Abbreviation | Description |
|---|---|---|
| Initiator | MCN_INIT | When a node decides to start a new session and announces it to the other nodes in the network, the node becomes a session initiator. |
| Server | MCN_SRV | When the first member joins the session and the initiator starts sending multicast data to this member, the initiator becomes a session server. |
| Predecessor | MCN_PRED | When a node receives the announcement for a new session from other intermediate nodes, it registers these nodes as its session predecessors. |
| Receiver | MCN_RCV | When a node decides to join another node's session and completes the necessary procedure to build a path between the session server and itself successfully, it becomes a session receiver. |
| Forwarder | MCN_FWD | When a node receives a message stating that it is on the selected path between a multicast sender and a receiver, the node becomes a session forwarder. |
| Forwarding receiver | MCN_FRCV | When a node is a receiver and also forwards data to others in a session, it is called a forwarding receiver. |

Table 3.2. Definitions for the tables in AQM

| Table name | Abbreviation | Description |
|---|---|---|
| Sessions | TBL_SESSION | Nodes keep track of the existing sessions by entering them in this table with the information they receive from the other nodes. |
| Members | TBL_MEMBER | Nodes use this table to maintain their relations with the other nodes in regard to each specific session. |
| Requests | TBL_REQUEST | Since nodes take part in the session joining processes of others, they have to keep track of the active join requests while they are handling them. |
| Neighbours | TBL_NEIGHBOUR | Each node has to know its neighbours in order to detect a possible loss of connection and be able to take the necessary actions. |

Table 3.3. Definitions for the messages in AQM

| Message name | Abbreviation | Description |
|---|---|---|
| Session initiation | SES_INIT | Nodes announce their new sessions by sending these messages, which are propagated by the others. |
| Session update | SES_UPDATE | Session initiators or servers periodically send these messages to inform new nodes of the existence of their sessions. |
| Session termination | SES_TERMINATE | Session initiators or servers send these messages in order to close the session they have started. |
| Session lost | SES_LOST | When nodes detect the loss of a neighbour, they have to check also whether they have lost their only connection to one or more sessions. In this case, they inform the others of the loss of the session using the session lost messages. |
| Session leave | SES_LEAVE | Nodes wishing to leave a session they have joined send these messages. |
| Join request | JOIN_REQ | Nodes wishing to join other nodes' sessions send these messages, which are propagated by intermediate nodes towards session members. |
| Join reply | JOIN_REP | Session members receiving join requests respond with these messages if they decide that they are capable of serving the request. |
| Join reserve | JOIN_RES | Nodes having sent join requests and received replies send these messages in return to reserve resources along the path they select. |
| Join error | JOIN_ERR | Nodes having sent join replies and received reserve messages respond with these messages if they are not capable of serving the request any more. |
| Neighbour greeting | NBR_HELLO | All nodes send periodic greeting messages, which are not propagated beyond their one-hop neighbours, to inform them of their existence. |

## 3.2. The Application Module

The application module of the AQM protocol is the interface between the user and the multicast routing system. Based on the decisions made by the user, this module initiates, terminates, joins and leaves sessions. Shortly after being activated, all AQM nodes become aware of the existing sessions they can join. Thus, a user can choose one of these sessions based on the application information provided by the session server, such as its type, duration, cost and QoS requirements. Alternatively, users can initiate their own sessions, whereby they set the QoS class and other preferences of the application, become a session server and wait for other users to join their session. Session servers are also responsible for streaming the multimedia contents in form of data packets they prepare.

### 3.2.1. Usage of QoS Classes

Different QoS classes are necessary to support various types of applications in an efficient manner. In any multimedia network, there may be multiple application types being run simultaneously, which need to be classified in terms of their varying QoS requirements. To represent such a generic networking environment in this thesis, a sample set of multimedia applications is assumed, which are defined by typical bandwidth requirements, end-to-end delay in terms of maximum allowable number of hops, average session and membership durations. Depending on the user profiles, network conditions and computational capabilities of the mobile multimedia devices, other applications with different QoS settings can easily be added to the set.

Another advantage of defining QoS classes is that it also limits the amount of information to be transmitted between network nodes. It is otherwise impossible to define and forward a best QoS combination without making some assumptions or disregarding some valuable data about the current QoS conditions being experienced by the network. Many times it is the case that a certain path offers the best conditions in terms of available bandwidth and packet loss probability while a different path with the shortest delay and minimum hop count values exists. Therefore, it may be preferable that nodes only inform others of the availability of a certain QoS support level and send updates only when this level changes.

### 3.3. The Session Module

The main function of the session module is the management of multicast sessions. Triggered by the application module, the session module generates and distributes session initiation and termination messages. It also handles similar messages received from other AQM nodes and delivered via the network module. Finally, it is the responsibility of the session module to maintain session integrity throughout the network by utilizing periodic session update and occasional session loss announcements.

Table 3.4. Data fields used in the table of sessions (TBL_SESSION)

| Field name | Description |
|---|---|
| Initiator | The node that initiates and announces the session. A node can initiate and serve only one session at a given time. A node joins a session by sending a request to its initiator. However, a request can also be replied by an intermediate node which is an active member of the session. |
| Identifier | A sequence number given by the session initiator to each new session it starts in an incremental manner. This is one of the ways how outdated join requests are distinguished from fresh ones. |
| QoS class | As the session initiator starts the application, the QoS class that it belongs to is selected, which automatically defines the bandwidth requirement and the hop count limitation of the session. |
| Termination | The termination time of the session. The average duration of a session depends on the type of the application. However, individual values may vary. With the help of this information, users can decide whether or not to join a specific session or process the join requests related to it. |
| Status | Nodes need a flag to rise when they forward a session initiation or update packet they have received. This is useful to prevent the unnecessary processing and forwarding of packets in case they receive another packet announcing the same session. |
| Timestamp | The time of the last modification on the record. Initially, it is the time of session initiation. The timestamp is refreshed after the reception of each new session update message. Intermediate nodes compare the difference between the timestamp and the current time with the session update timer and decide whether a session is still up-to-date or lost. |

The table of sessions (TBL_SESSION), which is introduced in Section 3.1, is used mainly by the session module. Each node maintains a session table, where it puts and updates session data acquired by the initiation, update, termination and loss packets mentioned above. The structure of the session table is given in Table 3.4.

The table of members (TBL_MEMBER), which is also introduced in Section 3.1, is used by the session module as well. In close relation with the table of sessions, each node also maintains a member table, where it puts and updates membership data. Whenever a node receives an initiation, update, termination or loss packet and revises the contents of its session table, it also creates or updates a record in the member table for the node that delivers the packet,. The structure of the member table is given in Table 3.5.

In addition to these two tables, four different types of control packets are utilized by the session module to manage the flow of information regarding the sessions between the mobile terminals. These are session initiation (SES_INIT), update (SES_UPDATE), termination (SES_TERMINATE) and loss (SES_LOST) packets.

Table 3.5. Data fields used in the table of members (TBL_MEMBER)

| Field name | Description |
|---|---|
| Session initiator | The node that initiates and announces the session. This field is used in conjunction with the following identifier field to keep the table of members in relation with the table of sessions. |
| Session identifier | The sequence number given by the session initiator to each new session it starts in an incremental manner. |
| Member | The identification of the node, such as its node number, which the table entry belongs to. The node may not be an active member of the session yet. Nevertheless, a record is created for each node that can later serve as one. |
| State | The actual role of the node in the session. The table owner's decisions regarding a session are based on the state of the members. |
| Hop count | The total number of hops between the session initiator and the table owner on the path via the predecessor that the table entry belongs to. |
| Timestamp | The time of the last modification on the record. |

### 3.3.1. Session Initiation

A session is started by a session initiator (MCN_INIT), which can be any node that broadcasts a session initiation packet (SES_INIT) consisting of the identification number and the QoS class of the new session, which sets the bandwidth and hop count rules to join it. If necessary, the session definition can be enhanced with the duration and the cost of the application, the minimum number of users to activate the session and the maximum number of acceptable users.

```
Read SES_INIT data received from application module;
Calculate neighbourhood bandwidth allocation;
Calculate available bandwidth;
IF (available bandwidth >= session bandwidth)
     {
     Send SES_INIT packet;
     Enter SES_INIT data into TBL_SESSION;
     Enter own node [state = MCN_INIT] into TBL_MEMBER;
     Set timer for first session update;
     }
Report initiation decision to application module;
```

Figure 3.2. Procedure for the initiation of a new session

Figure 3.2 shows the pseudocode of the session initiation procedure. According to this procedure, the actual command to send a SES_INIT packet comes from the application module. Upon the arrival of the command, the session module checks the availability of bandwidth to ensure that subsequent join requests can be accepted and necessary data flow can be maintained. At this phase it is not necessary to check the hop count limitation since a session is guaranteed to allow one hop. If the QoS conditions are met, the session module broadcasts the SES_INIT packet and makes the relevant entries to TBL_SESSION and TBL_MEMBER. It also schedules an interrupt by setting a timer for sending the first session update. Finally, the session module gives feedback to the application module about the session initiation.

A table of active sessions (TBL_SESSION) is maintained at each node to keep the information on session definitions. Using their session tables, nodes forward initiation packets of previously unknown sessions. A membership table (TBL_MEMBER) is used to denote the status of the predecessors (MCN_PRED) which have informed the node of the

existence of a particular multicast session and the QoS support status of the path from the session initiator up to this node via that predecessor. The hop count information in the packet is used to prevent loops in the forwarding process. The session initiation packet is forwarded as long as the QoS requirements are met. Before the packet is rebroadcast, each node updates its QoS information fields with the current QoS conditions experienced by that node. The packet is dropped if QoS requirements cannot be met any more.

```
Read SES_INIT packet received from network module;
Calculate neighbourhood bandwidth allocation;
Calculate available bandwidth;
IF ((new session) && (available bandwidth >= session bandwidth))
     {
     Enter SES_INIT data into TBL_SESSION;
     Enter packet sender [state = MCN_PRED] into TBL_MEMBER;
     Enter own node [state = MCN_PRED] into TBL_MEMBER;
     IF (own hop count <= session hop limit)
          {
          Forward SES_INIT packet;
          }
     }
ELSE IF ((known session) && (new predecessor))
     {
     Enter packet sender [state = MCN_PRED] into TBL_MEMBER;
     IF (new hop count < own hop count)
          {
          Update own node [new hop count] in TBL_MEMBER;
          }
     IF ((NOT forwarded SES_INIT packet) &&
        (available bandwidth >= session bandwidth)&&
        (own hop count <= session hop limit))
          {
          Forward SES_INIT packet;
          }
     }
```

Figure 3.3. Procedure for handling the received session initiations

Figure 3.3 displays the pseudocode for the handling of the SES_INIT packets by the intermediate nodes that receive them. These packets arrive at the session module of such a node via its network module. First, the packet receiver checks its TBL_SESSION to see whether this is a known session to it. In case this is a new session, the node also ensures that enough bandwidth is available to serve the session since, although they are neighbours with the packet sender, their neighbourhoods are not exactly the same and there are differences in the available bandwidth they can occupy. The node enters the session into TBL_SESSION and itself as well as the packet sender into TBL_MEMBER with the state

information MCN_PRED to denote that, if necessary, it can connect to the session initiator through that node. In case this is a known session reported by a new predecessor, the node enters the packet sender into TBL_MEMBER and updates its own hop count if there is an improvement. If the session initiation data has previously not been forwarded and the maximum allowed hop count for the session is not yet exceeded, SES_INIT is forwarded.

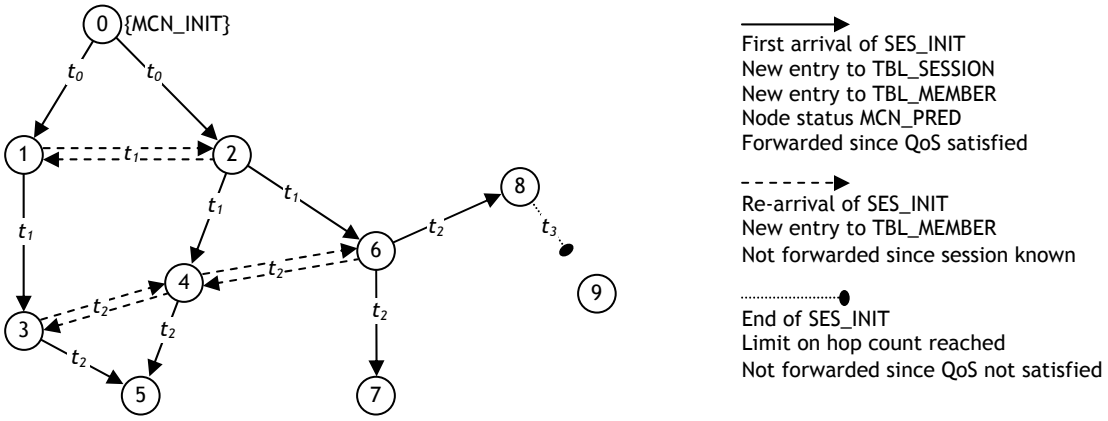



Figure 3.4. The AQM session initiation process

Figure 3.4 gives a demonstrative example for the phases of session initiation. SES_INIT is broadcast by MCN_INIT node $n_0$ for a new session. It propagates through the network with time, informing all the nodes from $n_1$ to $n_8$, which become MCN_PRED and update their TBL_SESSION and TBL_MEMBER. $n_9$ is not informed since it is beyond the QoS limits in terms of hop count. $t_i < t_{i+1}$, $0 \leq i \leq 3$, represent the relative timing of the packets. Upstream re-arrival arrows of SES_INIT are not shown to keep the figure simple.

### 3.3.2. Session Updates

After the initial announcement, the session information is refreshed periodically via session update packets (SES_UPDATE) sent by the session initiator. Similar to the session initiation packets, they are propagated throughout the network as long as the QoS requirements of the session can be fulfilled. Unlike the session initiation packets, however, each new update packet is forwarded once even if it belongs to a previously known session and comes from a known predecessor. This ensures that all new nodes in a neighbourhood are informed of the existence of the ongoing sessions they can join. A node in the process of sending session updates has to consider the possibility that the session cannot be

supported any more due to the changes in network topology or QoS conditions. In other words, it may be the case that a session initiator that has previously broadcast a session initiation packet does not have enough available bandwidth to send the application data if any node wishes to join its session. In that case, the network has to be informed that a previously announced session is temporarily unavailable. This is done by the initiator by omitting the session update message. All the nodes in the network set session update timers to check whether a session is being updated in a timely manner. If a node does not receive updates of a particular session for a time exceeding this timer, it considers that session lost and deletes all the related information from TBL_SESSION and TBL_MEMBER. Other details of lost session handling are given in Section 3.3.4.

```
Calculate neighbourhood bandwidth allocation;
Calculate available bandwidth;
IF ((own state == MCN_SRV) || ((own state == MCN_INIT) &&
   (available bandwidth >= session bandwidth)))
     {
     Send SES_UPDATE packet;
     }
Set timer for next session update;
```

Figure 3.5. Procedure for sending a session update

Figure 3.5 shows the pseudocode of the session update procedure. A session server sends its update directly since it already has receivers and resources are already allocated. However, an initiator has to check bandwidth availability. If QoS conditions cannot be met anymore, the initiator chooses not to send this SES_UPDATE. It can announce its session again at a later update if QoS improves.

Figure 3.6 displays the pseudocode for the handling of the SES_UPDATE packets by the intermediate nodes that receive them. If a node that does not know the session receives a SES_UPDATE, it treats the packet like a SES_INIT. If the session is known but the packet sender is new, it enters the new predecessor information into its TBL_MEMBER. If both the session and the predecessor are known, existing predecessor information is updated. The node also updates its own hop count if there is an improvement. Based on the availability of the necessary QoS conditions for the known session, SES_UPDATE is either forwarded or not.

```
Read SES_UPDATE packet received from network module;
Calculate neighbourhood bandwidth allocation;
Calculate available bandwidth;
IF ((new session) && (available bandwidth >= session bandwidth))
     {
     Enter SES_UPDATE data into TBL_SESSION;
     Enter packet sender [state = MCN_PRED] into TBL_MEMBER;
     Enter own node [state = MCN_PRED] into TBL_MEMBER;
     IF (own hop count <= session hop limit)
          {
          Forward SES_UPDATE packet;
          }
     }
ELSE IF ((known session) && (new predecessor))
     {
     Enter packet sender [state = MCN_PRED] into TBL_MEMBER;
     IF (new hop count < own hop count)
          {
          Update own node [new hop count] in TBL_MEMBER;
          }
     IF ((NOT forwarded SES_UPDATE packet) &&
        (available bandwidth >= session bandwidth)&&
        (own hop count <= session hop limit))
          {
          Forward SES_UPDATE packet;
          }
     }
ELSE IF ((known session) && (known predecessor))
     {
     Update predecessor [new hop count] in TBL_MEMBER;
     Get predecessors' [min hop count] from TBL_MEMBER;
     Update own node [min hop count + 1] in TBL_MEMBER;
     IF ((NOT forwarded SES_UPDATE packet) &&
        (available bandwidth >= session bandwidth)&&
        (own hop count <= session hop limit))
          {
          Forward SES_UPDATE packet;
          }
     }
```

Figure 3.6. Procedure for handling the received session updates

### 3.3.3. Session Termination

The session is closed by its initiator with a session termination message (SES_TERMINATE). Upon receiving the packet, all nodes knowing that session clean their tables, whereas nodes forwarding multicast data also free their resources allocated to the session. A node receiving a session termination packet forwards it if it is aware of the session as a predecessor or is currently forwarding session data to at least one active session member. Receivers of a terminated session are forced to leave the session.

```
Read SES_TERMINATE data received from application module;
IF ((own state == MCN_SRV) || ((own state == MCN_INIT) &&
   ((forwarded SES_INIT packet) || (forwarded SES_UPDATE packet))))
     {
     Send SES_TERMINATE packet;
     }
Delete session record from TBL_SESSION;
Delete all member records of session from TBL_MEMBER;
```

Figure 3.7. Procedure for the termination of a session

Figure 3.7 shows the pseudocode of the session termination procedure. Similar to the session initiation procedure, the command to send a SES_TERMINATE packet comes from the application module. The session module sends the packet if it is an active server with one or more receivers or it has previously announced its session. The module also clears all data related to this session from TBL_SESSION as well as TBL_MEMBER.

```
Read SES_TERMINATE packet received from network module;
IF (known session)
     {
     Forward SES_TERMINATE packet;
     Delete session record from TBL_SESSION;
     Delete all member records of session from TBL_MEMBER;
     }
```

Figure 3.8. Procedure for handling the received session terminations

Figure 3.8 displays the pseudocode for the handling of the SES_TERMINATE packets by the intermediate nodes that receive them. When a node is informed of the end of a session, it forwards the information if it is an active member of the session serving one or more receivers downstream. In case the node is only a receiver without any successors or it knows about the session but is not an active member of it, the node forwards the SES_TERMINATE even if it has not forwarded the corresponding SES_INIT or SES_UPDATE packets to make sure that no nodes attempt to join a terminated session. This behaviour may create a small amount of redundancy, which is justified by the need to adapt the dynamic topological conditions of a mobile ad hoc network. Finally, the session and membership records of TBL_SESSION and TBL_MEMBER are cleaned.

### 3.3.4. Session Losses

One of the major concerns for mobile ad hoc communications is the ability of the routing infrastructure to cope with the dynamics of node mobility. In order to maintain connectivity and support QoS with maximum possible accuracy and minimum overhead under mobility, nodes perform periodic cleanup operations on their session, membership and neighbourhood tables. If a node loses one of its neighbours, it is highly possible that the loss affects its connectivity to one or more sessions, be it an active member such as a server, a forwarder or a receiver, or just an ordinary predecessor merely aware of the session. Thus, a node has to inform its successors when it loses its connection to a session. This task is accomplished through the use of the lost session (SES_LOST) messages.

Figure 3.9 shows the pseudocode for the announcement procedure of lost sessions. If a node loses a neighbour which is serving it as a MCN_SRV or MCN_FWD in a particular session, it checks its TBL_MEMBER to see whether there are any other nodes connecting it to the session. If there are other MCN_FWD nodes or the MCN_SRV, then the node is not affected by the loss. If there are only MCN_PRED nodes, the node itself becomes a MCN_PRED. If there are no other nodes related to the session in any way, it deletes all data related to the session from its TBL_SESSION and TBL_MEMBER. If it is a forwarding member of the session or a receiver which has previously forwarded any SES_UPDATE packets announcing that session, it also informs its successors with a SES_LOST packet. If, on the other hand, a session server or a forwarding member loses a receiver, it updates its status to MCN_INIT, MCN_RCV or MCN_PRED depending on the existence of other receivers in that session. It does not need to inform other nodes of the lost session. However, it has to notify its predecessors that it leaves the session if its state changes from MCN_FWD to MCN_PRED as a result of losing its only receiver. Finally, if a MCN_PRED node loses its only predecessor for a specific session, it sends a SES_LOST to its successors and lets them know that it should be deleted from the list of predecessors for that session. Downstream nodes receiving the SES_LOST packet interpret them in a similar way to update their states regarding the lost session and forward the packet if necessary. This mechanism, combined with the periodic session updates mentioned previously, keeps nodes up-to-date regarding the QoS status of the sessions and prevents them from making infeasible join requests in terms of resource allocation.

```
Delete lost node from TBL_MEMBER;
Get predecessors' [min hop count] from TBL_MEMBER;
Update own node [min hop count + 1] in TBL_MEMBER;
IF (((lost node state == MCN_SRV) &&
     (NOT other nodes [state == MCN_FWD])) ||
     ((lost node state == MCN_FWD) &&
     (NOT other nodes [state == MCN_SRV])))
     {
     IF (other nodes [state == MCN_PRED])
          {
          Update own node [state = MCN_PRED] in TBL_MEMBER;
          }
     ELSE
          {
          Delete session record from TBL_SESSION;
          Delete all member records of session from TBL_MEMBER;
          }
     IF ((own node state == MCN_FWD) ||
        (own node state == MCN_FRCV) ||
        ((own node state == MCN_RCV) &&
        (forwarded SES_UPDATE packet)))
          {
          Send SES_LOST packet;
          }
     }
ELSE IF (lost node state == MCN_RCV)
     {
     IF (NOT other nodes [state == MCN_RCV])
          {
          IF (own node state == MCN_SRV)
               {
               Update own node [state = MCN_INIT] in TBL_MEMBER;
               }
          ELSE IF {own node state == MCN_FRCV}
               {
               Update own node [state = MCN_RCV] in TBL_MEMBER;
               }
          ELSE IF {own node state == MCN_FWD}
               {
               Update own node [state = MCN_PRED] in TBL_MEMBER;
               Send SES_LEAVE packet;
               }
          }
ELSE IF (((lost node state == MCN_PRED) ||
          (lost node state == MCN_INIT)) &&
          (own node state == MCN_PRED))
     {
     IF ((NOT other nodes [state == MCN_PRED]) &&
        (NOT other nodes [state == MCN_INIT]))
          {
          Delete session record from TBL_SESSION;
          Delete all member records of session from TBL_MEMBER;
          IF (forwarded SES_UPDATE packet)
               {
               Send SES_LOST packet;
               }
          }
     }
```

Figure 3.9. Procedure for the announcement of a lost session

The handling of the lost session information is an operation-intensive procedure. It is applied regularly by each node having detected the disappearance of a neighbour, whereby each session is examined with regard to whether it is related to the lost neighbour and the case requires further processing such as informing the successors. In contrast to the other procedures of the session module, which take worst-case running times $O(n)$, the lost session notifications yield an $O(n^2)$ upper bound. Other aspects of neighbourhood maintenance are explained is Section 3.5.1.

Figure 3.10 displays the pseudocode for the handling of the SES_LOST packets by the intermediate nodes that receive them. According to this procedure, there are three important session parameters that affect the decisions of the node receiving a SES_LOST. These parameters are: (i) The state of the packet sending node as known by the receiving node; (ii) The new state of the packet sending node as reported by itself in the packet; (iii) The own state of the packet receiving node regarding the session in question.

If the state of the immediate predecessor that has forwarded the SES_LOST packet is MCN_FWD according to the record of the receiver's TBL_MEMBER and this predecessor claims that it has become a MCN_PRED, this means that the receiver of the SES_LOST packet has just lost a session forwarder. It has to check whether it has other predecessors in the MCN_FWD state or it is served directly by a MCN_SRV, the session server. If it does, then the node does not have to do anything and can continue receiving multicast data from its remaining forwarders or server. If not, the node itself becomes a MCN_PRED and forwards the packet further downstream if its old state was either MCN_FWD or MCN_FRCV, which means that it has successors to inform as well.

If the state of the predecessor changes from MCN_FWD to NULL and the states of the other predecessors are neither MCN_SRV nor MCN_FWD, this means that the receiver of SES_LOST has lost its connection to the session completely. In that case, it deletes this predecessor from its TBL_MEMBER and looks for other session predecessors in order to determine its own new state. If it finds at least one other MCN_PRED in the table, it updates its own state to MCN_PRED. If there are no other predecessors at all, the node deletes the session from its TBL_SESSION. It forwards the SES_LOST packet if it has successors or been a receiver and forwarded the SES_UPDATE packet of this session.

```
Read SES_LOST packet received from network module
IF ((packet sender state == MCN_FWD) && (new state == MCN_PRED) &&
     ((NOT other nodes [state == MCN_FWD]) &&
     (NOT other nodes [state == MCN_SRV])))
     {
     Update packet sender [state = MCN_PRED] in TBL_MEMBER;
     IF ((own node state == MCN_FWD) || (own node state == MCN_FRCV))
          {
          Forward SES_LOST packet;
          }
     Update own node [state = MCN_PRED] in TBL_MEMBER;
     }
ELSE IF ((packet sender state == MCN_FWD) && (new state == NULL) &&
     ((NOT other nodes [state == MCN_FWD]) &&
     (NOT other nodes [state == MCN_SRV])))
     {
     Delete packet sender from TBL_MEMBER;
     IF (other nodes [state == MCN_PRED])
          {
          Get predecessors' [min hop count] from TBL_MEMBER;
          Update own node [min hop count + 1] in TBL_MEMBER;
          Update own node [state = MCN_PRED] in TBL_MEMBER;
          }
     ELSE
          {
          Delete session record from TBL_SESSION;
          Delete all member records of session from TBL_MEMBER;
          }
     IF (((own node state == MCN_RCV) &&
        (forwarded SES_UPDATE packet) &&
        (NOT other nodes [state == MCN_PRED])) ||
        (own node state == MCN_FWD) || (own node state == MCN_FRCV))
          {
          Forward SES_LOST packet;
          }
     }
ELSE IF ((packet sender state == MCN_PRED) && (new state == NULL))
     {
     Delete packet sender from TBL_MEMBER;
     Get predecessors' [min hop count] from TBL_MEMBER;
     Update own node [min hop count + 1] in TBL_MEMBER;
     IF (own node state == MCN_PRED)
          {
          IF ((NOT other nodes [state == MCN_PRED]) &&
             (NOT other nodes [state == MCN_INIT]))
               {
               Delete session record from TBL_SESSION;
               Delete all member records of session from TBL_MEMBER;
               IF (forwarded SES_UPDATE packet)
                    {
                    Forward SES_LOST packet;
                    }
               }
          }
     }
```

Figure 3.10. Procedure for handling the received lost session announcements

Finally, if the state of the node sending the SES_LOST packet is previously recorded as MCN_PRED by the receiver and the new state of the node is NULL according to the arriving packet, the receiver deletes the node from and updates its own record in its TBL_MEMBER first. Then it checks its own state for the session. If it is an active member of the session such as a forwarder or a receiver, it has to have a forwarder, which means that the SES_LOST packet does not need further processing. If, on the other hand, the receiver is a MCN_PRED itself, then it has to check whether it has other predecessors connecting it to the session. If this is not the case, the receiver deletes all related data from its TBL_SESSION and TBL_MEMBER. It also forwards the SES_LOST packet if it has previously forwarded a SES_ UPDATE for that session.

### 3.4. The Membership Module

As soon as a session is introduced to the network and the initial announcement is made, the nodes that become aware of the session are free to join it. Initiated actually by the application module, a join request message is prepared and broadcast by the membership module to build a path between the requesting node and the existing multicast graph. When it is time to leave a session, the application module of the session member triggers the membership module to send a notification-of-leave message upstream towards the predecessors of the node. The membership module also handles similar messages received from other nodes in the process of joining or leaving a session.

In addition to the session and member tables defined in Section 3.3, intermediate nodes maintain a temporary request table (TBL_REQUEST) to keep track of the requests and replies they have forwarded and prevent false or duplicate packet processing. Each request has an expiration time, at the end of which the request is considered rejected by the requesting node if no replies have been received. Expired requests are deleted from TBL_REQUEST. The fields of the request table are given in Table 3.6.

Five different types of control packets are utilized by the membership module to manage the flow of information regarding the join and leave processes between the mobile nodes. These are request to join (JOIN_REQ), reply (JOIN_REP), reserve (JOIN_RES), error (JOIN_ERR) and notification of leave (SES_LEAVE) packets.

Table 3.6. Data fields used in the table of requests (TBL_REQUEST)

| Field name | Description |
|---|---|
| Session initiator | The node that initiates and announces the session. This field is used in conjunction with the following identifier field to keep the table of members in relation with the table of sessions. |
| Session identifier | The sequence number given by the session initiator to each new session it starts in an incremental manner. |
| Requester | The identification of the node, such as its node number, which the table entry belongs to. |
| Expiration | The time of expiration for the request, at the end of which the request is considered rejected. |
| Forwarder | The immediate predecessor which the reply comes from. In case the requester chooses the path via the table owner, this information is used to trace the path back towards the first active session member which accepted the join request and originated the reply. |
| Status | Intermediate nodes have to keep track of the phase a particular join request has been through or is currently in, such as request forwarded, reply forwarded, reserve forwarded, or reserve error. |
| Timestamp | The time of the last modification on the record. |

### 3.4.1. Joining a Session

Joining a session is a three-phase process, which consists of the request, reply and reserve steps. Sometimes the reserve step may fail due to changes in the QoS conditions. In this case, a fourth step is involved, where the requesting node is informed of the error. The various phases of a join request are tracked by each intermediate node in temporary request tables (TBL_REQUEST).

Nodes can only join sessions known to them. When a node broadcasts a join request (JOIN_REQ) packet for a session, only upstream neighbours which are aware of the session take the request into consideration. These are predecessors of the requester and propagate the packet upstream as long as QoS can be satisfied. The upstream flow of the

request is guaranteed by comparing the hop count information of the packet with the distance to the server of the related session at intermediate nodes.

```
Read JOIN_REQ data received from application module
IF (own node state == MCN_FWD)
     {
     Update own node [state = MCN_FRCV] in TBL_MEMBER;
     Report successful join to application module;
     }
ELSE
     {
     Enter JOIN_REQ data into TBL_REQUEST;
     Send JOIN_REQ packet;
     Update process status [request forwarded] in TBL_REQUEST;
     Set expiration timer for JOIN_REQ;
     }
```

Figure 3.11. Procedure for the request to join a session

Figure 3.11 shows the pseudocode for initiating a join request. According to this process, a node can directly join a session if it is already an active member of that session. In other words, if the requesting node itself is a MCN_FWD, this means that, having made all the necessary arrangements such as resource allocation and route selection, it is already forwarding multicast data to one or more downstream members. Thus, the node becomes a MCN_FRCV without consulting any upstream nodes and the membership module returns a success report to the application module. Otherwise, the node issues a JOIN_REQ, puts the request data in TBL_REQUEST and sets a timer for the expiration of the request if no replies are received after a predefined timeout. The expiration timer can also be used such that all the collected replies are processed at the end of it.

A forwarded request eventually reaches some nodes which are already members of that session and therefore can directly send a reply (JOIN_REP) back to the requester. Members of a session are the initiator, the forwarders and the receivers. Downstream nodes, having forwarded join requests and waiting for replies, send these replies to the requesting node. Alternatively, they can also aggregate the replies they receive at the end of a predefined time period, select the one offering the best QoS conditions, combine it with the QoS they can currently offer and then send it towards the requester. The information about the originator and the immediate forwarder of the reply is kept in the packet.

```
Read JOIN_REQ packet received from network module;
IF ((known session) && (process status == NULL) &&
     (own hop count < packet sender hop count))
     {
     Calculate neighbourhood bandwidth allocation;
     Calculate available bandwidth;
     IF ((available bandwidth >= session bandwidth) &&
        (own hop count <= session hop limit) &&
        (own node state == MCN_PRED))
          {
          Enter JOIN_REQ data into TBL_REQUEST;
          Update process status [request forwarded] in TBL_REQUEST;
          Forward JOIN_REQ packet;
          Set expiration timer for JOIN_REQ;
          }
     ELSE IF (((own node state == MCN_SRV) ||
        (own node state == MCN_FWD) ||
        (own node state == MCN_FRCV)) ||
        (((own node state == MCN_INIT) ||
        (own node state == MCN_RCV)) &&
        (available bandwidth >= session bandwidth) &&
        (own hop count <= session hop limit)))
          {
          Enter JOIN_REQ data into TBL_REQUEST;
          Send JOIN_REP packet;
          Update process status [reply forwarded] in TBL_REQUEST;
          }
     }
```

Figure 3.12. Procedure for the reception of a join request

Figure 3.12 displays the pseudocode for the handling of the JOIN_REQ packets by the intermediate nodes. However, these packets are only handled on their first arrival if the session is known. If the intermediate node is not an active member of the session and the QoS conditions permit, it forwards the JOIN_REQ packet and sets the expiration timer for the new request. If, on the other hand, the node is a MCN_SRV already sending multicast data to its receivers or a member which has completed a previous join process successfully, such as a MCN_FWD or a MCN_FRCV, it can respond to the request with a JOIN_REP. Finally, a MCN_INIT waiting for its first receiver or a MCN_RCV which receives multicast data but does not forward them have to check for necessary QoS conditions since they have not allocated any resources yet. If QoS requirements can be met, they can respond to the request with a JOIN_REP. In all these cases, a record is entered to TBL_REQUEST to show the current request status.

```
Read JOIN_REP packet received from network module
IF ((process status == request forwarded) && (NOT expired JOIN_REQ))
     {
     Reset expiration timer for JOIN_REQ;
     IF ((packet sender state == NULL) &&
         (packet sender's hop count < own hop count))
          {
          Enter packet sender [state = MCN_PRED] into TBL_MEMBER;
          IF (packet sender's hop count + 1 < own hop count)
               {
               Update own node [packet sender's hop count + 1]
                                                    in TBL_MEMBER;
               }
          }
     IF (own request)
          {
          Update packet sender [role = upstream node] in TBL_REQUEST;
          Update packet sender [state = MCN_FWD] in TBL_MEMBER;
          Update own node [state = MCN_RCV] in TBL_MEMBER;
          Update process status [reserve forwarded] in TBL_REQUEST;
          Send JOIN_RES packet to packet sender;
          Report successful join to application module;
          }
     ELSE
          {
          Calculate neighbourhood bandwidth allocation;
          Calculate available bandwidth;
          IF ((available bandwidth >= session bandwidth) &&
             (own hop count <= session hop limit))
               {
               Update packet sender [role = upstream node]
                                                    in TBL_REQUEST;
               Update process status [reply forwarded]
                                                    in TBL_REQUEST;
               Forward JOIN_REP packet;
               }
          }
     }
```

Figure 3.13. Procedure for the reception of a join reply

Figure 3.13 shows the pseudocode for the handling of the JOIN_REP packets. Nodes also make use of the replies they receive during a session join process. If the reply is sent by a previously unknown node in response to a request it has forwarded for a session, the node enters that predecessor into its member table for future routing operations. If JOIN_REP is received by an intermediate node, it checks its TBL_REQUEST as a first step to see if it has forwarded a corresponding JOIN_REQ. If this is the case, the node has to forward the received JOIN_REP packet also. If the necessary QoS conditions exist, the node updates the record in its TBL_REQUEST and sends the JOIN_REP packet downstream towards the originator of the join request. As mentioned above, an alternate

procedure can be developed to make the intermediate nodes collect the replies they receive until the request expires and forward the one with the best QoS conditions to the requester.

When the JOIN_REP packet arrives at the originator of the JOIN_REQ, it changes its status from MCN_PRED to MCN_RCV and sends a JOIN_RES to the selected node that has forwarded the reply, which also becomes MCN_FWD in TBL_MEMBER of the requester. The status of the request is updated in TBL_REQUEST. At this point, the requester assumes that the join process is completed successfully and the membership module reports the result to the application module. If a JOIN_ERR is received later, this is also reported and appropriate action is taken. Again, an alternate procedure is possible whereby the requester postpones the path selection until the end of the JOIN_REQ expiration and then selects the reply with the best QoS conditions among possibly several JOIN_REP messages that it receives.

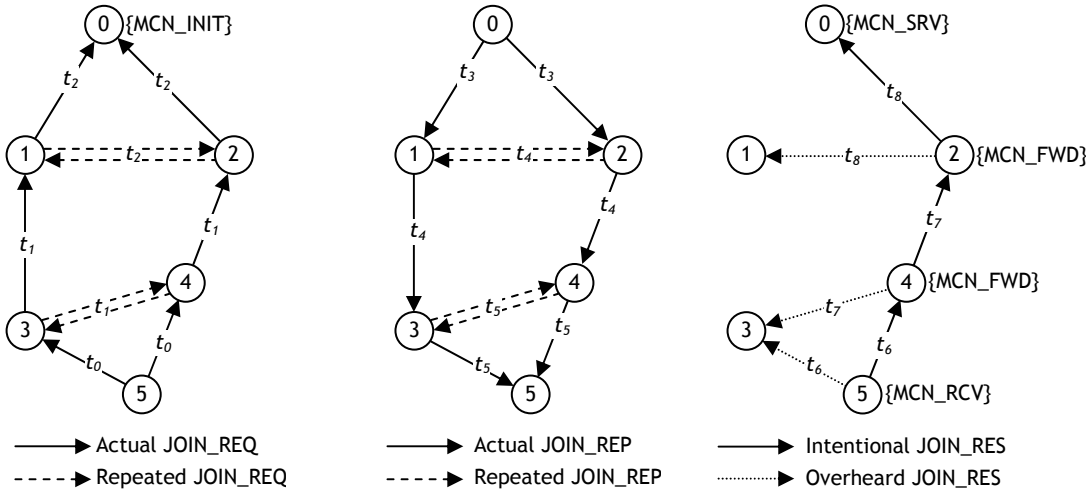
Figure 3.14 displays the pseudocode for handling the JOIN_RES packets. Upon receiving it, intermediate nodes check whether they are among the intended forwarders on the path from the selected replier towards the requester. If this is the case and they can still afford the resources for the session, they change their status from MCN_PRED to MCN_FWD, reserve resources and update their TBL_MEMBER to keep the list of their forwarders and successors for that session up-to-date. They forward the JOIN_RES packet upstream. Eventually, the reserve message reaches the originator of the reply, which can be a MCN_INIT with some or without any members, a MCN_FWD with one or more successors, or a MCN_RCV. If the replier is the session initiator and this is its first member, it changes its status to MCN_SRV. If it is a receiver, it becomes a MCN_FWD. In both cases, the replier records its successor in its TBL_MEMBER and reserves resources to start sending multicast data. If the node is an active server or forwarder, it must have already reserved resources. It only adds the new member to its TBL_MEMBER and continues sending the regular multicast data.

Intermediate nodes having sent a reply to a join request do not actually reserve resources until they receive the corresponding reservation packet. Since it is not certain that they are on the best QoS path according to the requester, they wait for this confirmation before activating that reservation and updating their resource availability

data. Therefore, it is possible that a node receives simultaneous join requests for other sessions before having made that final reservation for an ongoing join process and replies the new requests although the cumulative QoS requirements cannot be satisfied anymore. If multiple requesters select the path via the common replier as their path to join their multicast sessions and send their reservation packets in that direction, the node grants only the reservation that arrives first and sends reservation error messages (JOIN_ERR) to the others to let them know that the final reservation of resources fails for them. In this case, the failing requesters select the next best replies from their request tables and try these alternatives by sending their reservation messages to the nodes which have sent them.

```
Read JOIN_RES packet received from network module
IF (process status == reply forwarded)
     {
     Calculate neighbourhood bandwidth allocation;
     Calculate available bandwidth;
     IF ((available bandwidth >= session bandwidth) &&
        (own hop count <= session hop limit))
          {
          IF (own node state == MCN_PRED)
               {
               Get upstream node from TBL_REQUEST;
               IF (upstream node state == MCN_INIT)
                    {
                    Update upstream node [state = MCN_SRV]
                                                    in TBL_MEMBER;
                    }
               ELSE
                    {
                    Update upstream node [state = MCN_FWD]
                                                    in TBL_MEMBER;
                    }
               Update own node [state = MCN_FWD] in TBL_MEMBER;
               Forward JOIN_RES packet to upstream node;
               }
          ELSE IF (own node state == MCN_INIT)
               {
               Update own node [state = MCN_SRV] in TBL_MEMBER;
               }
          ELSE IF (own node state == MCN_RCV)
               {
               Update own node [state = MCN_FRCV] in TBL_MEMBER;
               }
          Update packet sender [state = MCN_RCV] in TBL_MEMBER;
          }
     ELSE
          {
          Send JOIN_ERR packet to packet sender;
          }
     }
```

Figure 3.14. Procedure for the reception of a join reserve




(a) Request phase (b) Reply phase (c) Reserve phase

Figure 3.15. The AQM session joining process

Figure 3.15 gives a demonstrative example for the phases of joining a session. (a) JOIN_REQ is issued by $n_5$. It propagates towards any member of the session as long as QoS can be satisfied. Nodes from $n_1$ to $n_4$ update their TBL_REQUEST as they forward the packet since they are not session members. (b) JOIN_REP is sent back from $n_0$ to $n_5$. It is forwarded by $n_1$, $n_2$, $n_3$, $n_4$. (c) $n_5$ sends JOIN_RES along the selected QoS path via $n_4$, $n_2$, $n_0$, which reserve resources and update their status. Other nodes ignore the message. $t_i < t_{i+1}$, $0 \le i \le 8$, represent the relative timing of the messages.

Each time a request-reply-reserve process completes successfully, intermediate nodes gather enough routing and membership data to take part in the packet forwarding task. When a host sends multicast packets for a particular multicast session, its neighbours already know if they are involved in the session by checking the two tables related to each other, one with information on their own membership status and another with a list of multicast sessions they are responsible of forwarding.

### 3.4.2. Adding Extra Forwarders

One of the major concerns about mobile ad hoc networks is the fact that they have to operate in an environment of continuous topological changes. The mobility and the limited transmission range of nodes cause their wireless links to break very frequently, leading to

disconnections and data loss. Therefore, it is particularly important that AQM increases its robustness. This can be done if extra forwarders can be added to the initial multicast tree, evolving it to a multicast mesh during the course of the data streaming phase in an intelligent way such that member connectivity is strengthened without compromising efficiency in resource usage. AQM makes use of the inherent broadcast capability of the ad hoc network to achieve this goal.

```
IF (packet sender state == MCN_INIT)
     {
     Update packet sender [state = MCN_SRV] in TBL_MEMBER;
     }
ELSE IF (packet sender state == MCN_PRED)
     {
     Update packet sender [state = MCN_FWD] in TBL_MEMBER;
     }
ELSE IF (packet sender state == NULL)
     {
     Enter packet sender [state = MCN_FWD] into TBL_MEMBER;
     Update packet sender [packet's hop count - 1] in TBL_MEMBER;
     IF (packet's hop count < own hop count)
          {
          Update own node [packet's hop count] in TBL_MEMBER;
          }
     }
Send JOIN_RES packet to packet sender;
```

Figure 3.16. Procedure for the addition of an extra forwarder during data reception

Figure 3.16 shows the pseudocode for the addition of a new forwarder to the multicast graph by a receiver. If a MCN_RCV, a MCN_FRCV or a MCN_FWD starts overhearing multicast data packets from a new node, which is a predecessor or a previously unknown node according to its TBL_MEMBER, this means that there is actually a new active session member within its transmission range other than its selected MCN_FWD for the session. In this case, the receiver decides that it can use this new node streaming multicast data actually to another node as an extra forwarder in order to improve its chance of remaining connected to the session despite frequent topological changes. The receiver registers this additional forwarder as a new MCN_FWD in its TBL_MEMBER for that session. It also informs the new forwarder of its existence to ensure that the forwarder is also aware of its new receiver. In addition to increasing the probability of connectivity and decreasing the frequency of reconnection attempts for the node itself, this operation is also useful for other receivers further downstream in case the node is not just a

MCN_RCV but a MCN_FRCV or a MCN_FWD since it improves their chances of connectivity as well. As a result of this, session satisfaction is increased in general. Moreover, since the robustness achieved through the addition of extra forwarders leads to fewer session losses, the operation also yields to less control overhead.

```
Read JOIN_RES packet received from network module;
IF (packet sender state != MCN_RCV)
     {
     Update packet sender [state = MCN_RCV] in TBL_MEMBER;
     }
```

Figure 3.17. Procedure for the addition of an extra receiver during data transmission

Figure 3.17 displays the short pseudocode for the handling of a received JOIN_RES packet by an active session member such as a MCN_SRV, MCN_FWD or MCN_FRCV forwarding multicast data packets. Upon reception, the node adds the packet sender to the group of receivers regarding that session. This operation ensures that even if the forwarder loses all of its previously-known receivers, it still continues to forward data packets to the one which has registered itself directly through the data streaming process.

The addition of extra forwarders during the data streaming phase has several benefits. First of all, forwarders and receivers are connected to their session servers through multiple intermediate forwarders, which makes session losses a less frequent issue. In other words, by increasing redundancy in their graphs, multicast sessions become less prone to lost forwarders. Secondly, a receiver adds a new forwarder only if it delivers fresh data packets, i.e., the extra forwarders create new paths for the receivers which are shorter than they have initially selected. These improvements lead to increased data delivery rates and decreased end-to-end delay, which means that more session members are satisfied by the provided QoS. Moreover, the addition of the extra forwarders does not generate extra data traffic since they are already forwarding data to their existing receivers.

### 3.4.3. Leaving a Session

The decision to leave a session is made by the application module, which triggers the membership module. A node needs to inform its forwarders on the multicast graph upon

leaving a session. After receiving the quit notification (SES_LEAVE), the forwarding nodes delete the leaving member from their member tables. If this has been its only successor in that session, a forwarding node checks its own status regarding the session. If the forwarder itself is also a receiver, it updates its status. Otherwise, it frees resources and notifies its predecessors of its own leave.

```
Read SES_LEAVE data from application module
IF (own node state == MCN_RCV)
     {
     Update own node [state = MCN_PRED] in TBL_MEMBER;
     Send SES_LEAVE packet;
     }
ELSE IF (own node state == MCN_FRCV)
     {
     Update own node [state = MCN_FWD] in TBL_MEMBER;
     }
```

Figure 3.18. Procedure for leaving a session

Figure 3.18 shows the pseudocode for leaving a session. A node can send the SES_LEAVE packet only if it is a MCN_RCV and does not have any successors that it serves. If it does, it changes its state from MCN_FRCV to MCN_FWD but has to remain a session member.

```
Read SES_LEAVE packet from network module
IF (packet sender state == MCN_RCV)
     {
     Delete packet sender from TBL_MEMBER;
     IF (NOT other nodes [state == MCN_RCV])
          {
          IF (own node state == MCN_FRCV)
               {
               Update own node [state = MCN_RCV] in TBL_MEMBER;
               }
          ELSE IF (own node state == MCN_SRV)
               {
               Update own node [state = MCN_INIT] in TBL_MEMBER;
               }
          ELSE
               {
               Update own node [state = MCN_PRED] in TBL_MEMBER;
               Send SES_LEAVE packet;
               }
          }
     }
```

Figure 3.19. Procedure for handling the received session leave notification

Figure 3.19 displays the pseudocode for handling the received SES_LEAVE packets. Upon receiving the notification, the node checks if it has other receivers. If it does not, it means that the node has lost its only or last receiver regarding the session. When this is the case, a MCN_FRCV becomes a MCN_RCV, a MCN_SRV becomes a MCN_INIT and a MCN_FWD becomes a MCN_PRED. The last possibility implies that the node does not have to be actively connected to the multicast group. Thus, it sends a SES_LEAVE of its own to inform its predecessors that it is no longer required as a multicast member.

## 3.5. The Network Module

The network module of the AQM protocol is the interface between the multicast routing system and the wireless communication infrastructure. The most important service provided by this module is the transmission and reception of packets. On the other hand, it is also the responsibility of the network module to classify the packets it receives from other nodes and deliver them to either the session or the membership module. Finally, the table of neighbours (TBL_NEIGHBOUR), which is briefly explained in Section 3.1, is maintained by the network module. Each node has a neighbour table, where it puts and updates neighbour data acquired by the greeting messages received. Since nodes also exchange their bandwidth allocation data with these messages, the information aggregated in the table of neighbours is used by the session and membership modules of the node to calculate the total bandwidth reservation within a neighbourhood and derive the available bandwidth for future routing operations. The details of these calculations are presented in Section 4. The structure of the neighbour table is given in Table 3.7.

Table 3.7. Data fields used in the table of neighbours (TBL_NEIGHBOUR)

| Field name | Description |
|---|---|
| Neighbour | The node that periodically sends greeting messages to the table owner. |
| Bandwidth | Current bandwidth requirement of the neighbour concerning the sessions. |
| Status | Recent history of the node, e.g., new, lost, found or deleted. |
| Timestamp | The time of the last greeting message received from the neighbour. |

In order to populate this table, another type of control packet is utilized by the network module, which is the neighbour greeting (NBR_HELLO) message.

### 3.5.1. Neighbourhood Maintenance

Each node periodically broadcasts greeting messages (NBR_HELLO), informing its neighbours of its existence as well as its bandwidth allocation, which is determined by the QoS classes of the sessions being served or forwarded by that node. Greeting messages can be piggybacked to other control and data messages to reduce control overhead. In other words, nodes do not need to send greeting messages explicitly unless they have not sent any piggybacked greeting messages for a certain period of time. Each node aggregates the information it receives with these messages in its neighbourhood table (TBL_NEIGHBOUR). Thus, the table shows the existence and current bandwidth allocation of all neighbours of a node. This table is used to calculate the total bandwidth currently allocated to multicast sessions in the neighbourhood, which is the sum of all the allocated capacities of the neighbouring nodes for that time frame. Neighbourhood tables also help nodes with their decisions on packet forwarding. Session initiation packets are forwarded only if a node has neighbours other than its predecessors for that session.

If a node does not receive any greeting messages from a neighbour for a while, it considers that neighbour lost. Lost neighbours are kept in the neighbourhood table for a predefined short amount of time, at the end of which they are deleted if they do not reappear. To prevent unnecessary message exchanges, nodes need to detect new neighbours quickly and distinguish them from reappearing lost neighbours, which do not necessitate any update. If the deleted neighbour is related to a session, it is also removed from the session, membership and request tables. When this is the case, additional action can be necessary depending on the state of the deleted neighbour as well as that of the node itself. An important task related to the handling of lost neighbours is the announcement of lost sessions, which is explained in greater detail in Section 3.3.4. This is an essential operation to keep the nodes up-to-date regarding the sessions and ready for future membership management activities such as initiating new join requests or replying to other nodes' join requests.

### 3.6. Comments on Implementation

As mentioned previously, research and development are taking place to define the next generation of wireless broadband multimedia communication systems. The global multimedia network of the future will probably consist of a fixed network with a wired backbone, an infrastructured mobile network with base stations and, at the peripherals, ad hoc mobile networks, which will be connected to the main internetwork via ad hoc switches. A global solution for a future internetwork has to consist of solutions for each type of network and mechanisms for integrating these solutions. As far as multicast sessions are concerned, the goal is to provide a seamless service whereby a single multicast group can span both network types, which requires the design of a gateway between the ad hoc and the wired networks [2].

Since AQM is a special-purpose, QoS-aware multicast routing protocol, it provides additional functionality to a portable computing device which has the basic radio transmission and reception capabilities. Such a device is typically equipped with wireless communication hardware and software such as the wireless LAN (WLAN) standard known as IEEE 802.11 [85]. Thus, it can reach an access point and connect to the WLAN infrastructure in order to communicate with other nodes in the WLAN domain or even in the wired portion of the LAN and also access the Internet. Thus, different situations are possible regarding the real-life scenarios mentioned above, three of which are summarized in the following paragraphs. Although AQM is a stand-alone protocol which is designed for ad hoc QoS multicast routing only, an AQM-enabled portable computing device should preferably be capable of handling these situations.

First of all, it is possible that an AQM-enabled device enters the WLAN domain. In a typical campus environment, the wired backbone network is extended by infrastructured wireless coverage via access points. Thus, AQM-enabled devices can communicate with each other in an ad hoc fashion when they are outside the WLAN domain. However, they can also reach the backbone if they enter the coverage area of an access point. When an infrastructured wireless network is available, the AQM-enabled device should be able to communicate with the access point and register itself with the WLAN in order to benefit from the services offered. This requires that the device can switch to the conventional

transmission control protocol/Internet protocol (TCP/IP) stack with WLAN support at the data link layer. Thus, it is possible that an AQM-enabled device can still connect to the LAN/WLAN with the help of the access point even if it is out of reach of the other AQM-enabled devices.

Secondly, it is possible that an ordinary computer in the LAN/WLAN domain wishes to join an AQM session. As explained previously, an AQM-enabled device initiates and announces a multicast session which is propagated through the ad hoc network. When the announcement reaches an access point with the AQM gateway functionality, necessary conversions have to be made in order to propagate the AQM session through the LAN/WLAN domain. Similarly, AQM session update, termination and join messages have to be converted to and from the existing Internet multicast protocol at the intersection point of the wired and ad hoc domains. One of the important tasks with regard to this process is that the gateway receiving the session initiation message from an AQM-enabled device has to associate the AQM session with a multicast address which can be identified in the LAN/WLAN domain such that connections can be established between wired and ad hoc devices.

Finally, it is also possible that an AQM-enabled device joins a multicast session which is initiated by a non-AQM-enabled device in the LAN/WLAN domain. It is preferable to let the AQM gateways deal with the protocol conversion tasks in order to enable such a session. Internet multicast requires specific mechanisms to track group membership, such as the Internet Group Management Protocol (IGMP) [86] and Multicast Listener Discovery (MLD) for [87]. With both protocols, routers periodically send query messages and receive reports of all multicast addresses being listened to by local hosts. Although the conversion procedure is mainly the reverse of the one explained above, the sessions initiated on the wired part of the network need to be announced on the ad hoc part according to the rules of AQM. In other words, the use of AQM as a multicast routing protocol only makes sense if the management of the session on the LAN/WLAN part of the network is based on the same QoS rules. Since the multicast strategy adopted by the LAN/WLAN may be completely different from AQM, it is necessary to develop a mechanism to be implemented at the gateway, which decides whether a LAN/WLAN session is AQM-compatible and announces only those sessions that comply with this rule

to the ad hoc portion of the network. If the necessary conversion functionality cannot be implemented at the gateways, this task is shifted to the last AQM-enabled ad hoc device communicating with it. In this case, the device is responsible to convert AQM control messages to the Internet multicast protocol control messages to be processed in the LAN/WLAN domain.

Seamless integration of ad hoc routing protocols with the existing Internet infrastructure has previously been addressed in the literature. An example of these efforts is the testbed implemented for the dynamic source routing (DSR) protocol, which operates at the IP layer and permits interoperation between different physical network interfaces [88]. Its implementation conceptually operates as a virtual link layer just under the normal IP layer. There is a gateway to connect nodes of the ad hoc network to nodes outside the network. There are also other mobile nodes, which use DSR when they realize that they are in the range of the DSR network but use mobile IP at the same time to enable packets from outside the DSR network to reach them. They also use mobile IP to connect to the Internet. Similar to the mobile IP scheme, all nodes have a unique IP address called their home address. The route discovery mechanism is extended to support communication between nodes inside the ad hoc network and those outside in the Internet. If a route request reaches the gateway node, which knows that the destination is reachable outside the ad hoc network, it sends a proxy reply listing itself as the second-to-last node in the route.

A further testbed example deals with the implementation and validation of the ad hoc multicast protocol ODMRP [89]. In this work, ODMRP control packets are implemented as new types of IGMP packets which include a data section. The existing IGMP packet structure and handler function are expanded to include control functionality. Multicast backbone traffic is forwarded from the wired to the wireless network via gateways and routers that use the distance vector multicast routing protocol (DVMRP), which routes multicast datagrams through the Internet. The streaming of the multicast traffic in the wireless testbed is realized by ODMRP and DVMRP routers. A hybrid router that converts DVMRP multicast feed into ODMRP-ready multicast can be developed.

Today's WLAN and mobile Internet users expect access to the services available in wired networks, including multimedia applications. Many efforts are being made to provide efficient mobility and multicast support and bring the two together in the next generation networks [90]. On the other hand, the diversification of today's networks yields ever more heterogeneity with various types of terminals, whereas protocols are mainly designed for one type only. In addition, node mobility is not considered in wired network protocols. Therefore, it is preferable to adapt an ad hoc protocol to the wired domain and not vice versa. What are necessary are a dual protocol stack and a translation mechanism to exchange control messages between the protocols [91]. When there are wired and wireless devices together in a multicast group, a gateway node is needed in between for translation. This node should have a low failure probability as well as mobility and be close to the access point of the WLAN. The translation function can be implemented at each ad hoc node or at a wired node that can be reached by the ad hoc nodes via a rendezvous point.

# 4. ESTIMATION OF BANDWIDTH REQUIREMENTS

The nodes in an ad hoc network have to maintain their resource information with as much accuracy as possible to support QoS, which includes the ability to keep track of available bandwidth within their neighbourhood. AQM nodes present a proactive behaviour with regard to management of the multicast session information by maintaining routing tables. They keep themselves and their neighbours aware of the changes in the QoS conditions and node connectivity regarding the multicast sessions known to them. The rationale behind this method is that QoS management in a highly dynamic environment such as mobile ad hoc networks cannot be achieved satisfactorily without informing the network of these issues in advance.

Ad hoc networks are highly dynamic and available resources may change considerably after the arrival of the QoS conditions with the first session initiation packet. Therefore, the nature of a join process is on-demand. AQM checks the most up-to-date QoS conditions during this process. Greeting messages are exchanged between neighbours to update nodes on the bandwidth allocation within a neighbourhood. This is how the nodes are able to provide their neighbours with valid routes when asked to take part in a request-reply-reserve process of another node wishing to join a multicast session.

## 4.1. Resource Allocation

The estimation of the available bandwidth is an important task for AQM. As shown in the pseudocode presented in Section 3, the procedures of AQM check the bandwidth availability for QoS purposes whenever a node has to decide whether or not to take part in multicast activities. Therefore, it is necessary to clarify the method developed for the calculation of the available bandwidth. Due to the broadcasting nature of the wireless medium, residual capacities are node-based, i.e., a node's available bandwidth is the residual capacity in its neighbourhood. In an ideal model, it is assumed that the bandwidth of a link can be determined on its neighbouring links [28]. Thus, each node calculates its current bandwidth allocation by aggregating the bandwidth requirements of the multicast sessions it takes part in as a server or forwarder.

**Definition 4.1:** Let $N$ = { network nodes }, *the set of the nodes* in the ad hoc network and $S$ = { multicast sessions }, *the set of the multicast sessions*. Let $S_i \subseteq S$ denote *the subset of sessions* that have node $i \in N$ on their multicast graph as a data forwarding node. Let $b_{s_k}$ be *the bandwidth requirement* of any session $s_k \in S_i$ served by node $k \in N$.

The total bandwidth allocation $b_i$ of node $i$ can be formulated as follows:

$$b_i = \sum_{s_k \in S_i} b_{s_k} \tag{4.1}$$

After making this calculation, each node informs its neighbours of the total amount of bandwidth it allocates to ongoing sessions via periodic greeting messages, which are introduced in the preceding section. The reserved bandwidth in a neighbourhood is the sum of the capacities allocated by all the nodes in that neighbourhood.

**Definition 4.2:** Let $r$ be *the transmission range* of the nodes in the ad hoc network and $d_{ij}$ *the Euclidian distance* between two nodes $i, j \in N$. Let $N_i = \{ j: d_{ij} \leq r \}$, $N_i \subseteq N$, *the set of the neighbours* of $i$, which is also defined as *the neighbourhood* of node $i$.

Being informed of the bandwidth allocation of all its neighbours, node $i$ calculates $b_{N_i}$, the total bandwidth allocation of its neighbourhood as follows:

$$b_{N_i} = \sum_{j \in N_i} b_j \tag{4.2}$$

Thus, each node derives the remaining bandwidth $\beta_i$ available in its neighbourhood, which is the maximum amount of bandwidth that it can allocate to new join requests, by subtracting the reserved bandwidth $b_{N_i}$ from the maximum bandwidth $\beta$ provided by the wireless medium:

$$\beta_i = \beta - b_{N_i} \tag{4.3}$$

It is noteworthy that for each node, the result of Equation 4.1 changes whenever the node gets an active role in a new session or is released by an ongoing session, whereas the results of Equation 4.2 and Equation 4.3 can change with every greeting message from a new or existing neighbour. Thus, $b_i$, $b_{N_i}$ and $\beta_i$ have to be recalculated each time the node is about to decide on accepting or rejecting a new session join request. On the other hand, these recalculations are not necessary if the new request is for a session $s_k \in S_i$, which the node is already serving and has already checked and allocated the necessary resources.

## 4.2. Virtual Tunnel of Bandwidth

Although neighbours keep each other informed of their current bandwidth allocations through the use of the periodic greeting messages, a node has to decide promptly whether or not to take part in a session in response to another node's join request based on the information at hand. This information may not be accurate enough considering the fact that the decisions, which lead to a multi-hop path from the server to the requester, are made by a string of nodes one by one without sending extra greeting packets or any other messages carrying updated information to each other. Thus, a node has to evaluate the availability of resources not only for itself but for all the members of the string that are within its range. It has to check whether, once selected by the requester as a forwarder for the multicast session, it can afford the bandwidth needed to support the streaming of the multimedia data of a certain QoS class through a multi-hop connection. In other words, it has to take the continuous flow of data in multimedia applications into account.

When a session initiator, which has announced a session of a certain QoS class and is waiting for its first member, receives a join request, it checks whether enough bandwidth is available to start sending multicast data by comparing the residual bandwidth of the neighbourhood to the requirement of its application. The problem arises when the residual bandwidth is satisfactory for the initiator to start serving data, but not enough for the immediate downstream neighbour to forward it towards the requester. Since the data is a multimedia stream, both nodes need bandwidth continuously. The problem is the same for the other predecessors downstream which are about to become forwarders.

An active session server or a forwarding intermediate member should not reply a new join request automatically although they have allocated necessary resources and are serving one or more receivers already. The arrival of the new request means that a new downstream node is about to join the multicast graph, which will start forwarding packets and consume additional resources. Therefore it is not enough that a node is already forwarding packets for a session to decide on supporting the new request for the same session since it may be the case that the new forwarder cannot find available resources.

In order to prevent overload, nodes have to ensure that once they accept being a forwarding member of a session, the available bandwidth within their neighbourhood is also enough for their predecessors and successors. In other words, a node has to forward a join request or a reply only if the residual bandwidth in its neighbourhood will also be enough for the immediate forwarders of the node, both upstream as well as downstream, in case the node will be on the multicast path chosen by the requester. The streaming nature of multimedia applications requires such a pipelined approach to checking bandwidth availability, which is termed the virtual tunnel of bandwidth in the following paragraphs.

Figure 4.1 shows a virtual tunnel of bandwidth between the server and the receiver. At each hop on the path, the sending and receiving nodes of the multicast data have to consider the issues mentioned above, e.g., $n_0$ sends packets and lets $n_1$ forward them, whereas $n_1$ first receives, then sends packets and finally lets $n_2$ forward them to $n_3$.

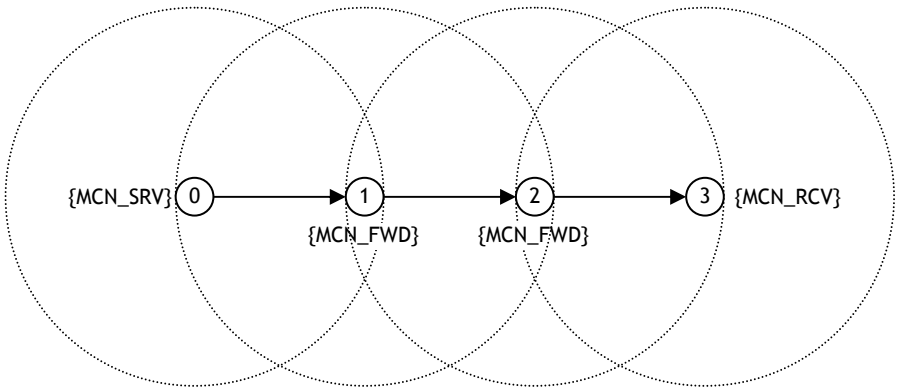



Figure 4.1. The virtual tunnel of bandwidth for the multicast members

**Definition 4.3:** Let $F_{i,s_k}$ = { downstream neighbours of *i* and forwarders in $s_k$ }, $F_{i,s_k} \subseteq N_i$, *the set of the downstream forwarders* that are within the neighbourhood of node *i*, which take part in the data forwarding process of $s_k \in S_i$. Let $\phi_{i,s_k} = \left|F_{i,s_k}\right|$, *the cardinality* of $F_{i,s_k}$.

Depending on the value of $\phi_{i,s_k}$ and the position of node *i* on the multicast graph of $s_k$, the bandwidth $q_{i,s_k}$ that is necessary in the neighbourhood of *i* to satisfy the QoS requirements of $s_k$ and support the continuous flow of the data packets for the session is calculated as follows:

$$q_{i,s_k} = \begin{cases} (\phi_{i,s_k} + 1) b_{s_k} & , \text{ if } i = k \\ (\phi_{i,s_k} + 2) b_{s_k} & , \text{ otherwise} \end{cases} \tag{4.4}$$

where node *k* is the server of the session.

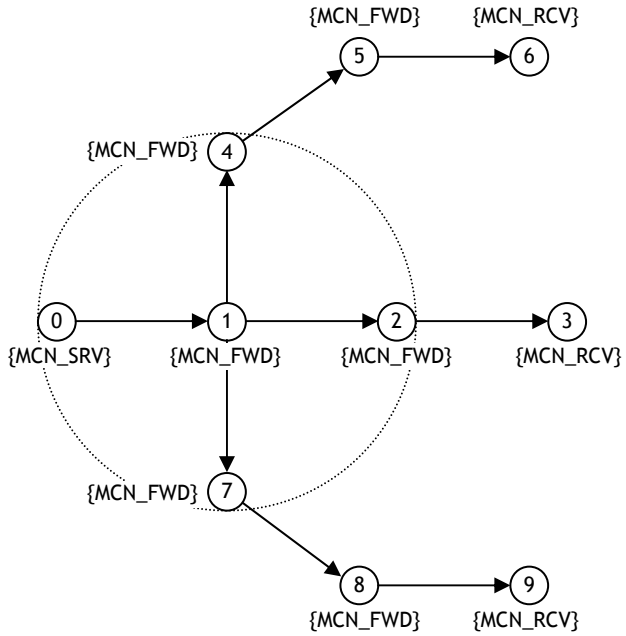



Figure 4.2. The multicast graph of $s_0$ and the neighbourhood of $n_1$

Figure 4.2 shows a directed multicast graph rooted at $n_0$ to demonstrate the calculation of $q_{1,s_0}$, the required QoS bandwidth from the viewpoint of the intermediate forwarder $n_1$ for

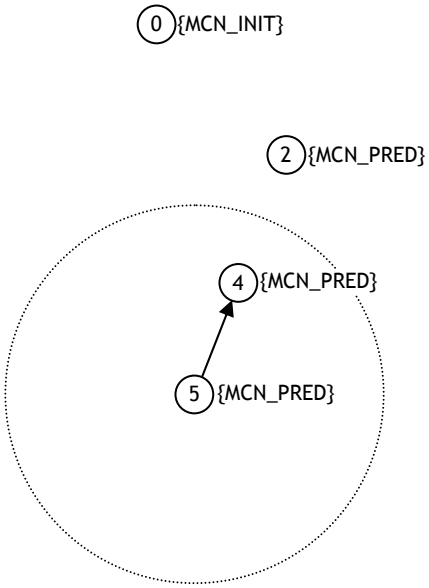
a session $s_0$. In the example, $n_1$ has three downstream forwarders in this session, $n_2$, $n_4$ and $n_7$. Thus, $\phi_{i,s_k}$ is equal to three and the total amount of bandwidth which has to be available in the neighbourhood of $n_1$ is five $b_{s_k}$ according to Equation 4.4. This is the sum of the individual bandwidth requirements of $n_0$, $n_1$, $n_2$, $n_4$ and $n_7$. First, $n_1$ receives packets from $n_0$, which requires $b_{s_k}$. Then it forwards them to $n_2$, $n_4$ and $n_7$ in a broadcasting manner, which requires another $b_{s_k}$.Finally, it allows each of these three downstream nodes to send the packets further downstream, which requires three $b_{s_k}$.

Concerning a session initiator about to allocate resources for its first member, twice as much bandwidth has to be available in the neighbourhood than the amount required by the QoS class of the session. There is only one forwarding node immediately following the server on the path to the member, which naturally belongs to the same neighbourhood as the initiator. Being within the transmission range of each other, they share the bandwidth of the same neighbourhood. Therefore, a session server has to ensure that its successor also has enough bandwidth available to forward multicast data packets that it receives. For the general case, the server requires $b_{s_k}$ for sending its own packets; and an additional $\phi_{i,s_k} b_{s_k}$ have to be available such that each of its $\phi_{i,s_k}$ successors are able to forward these packets.

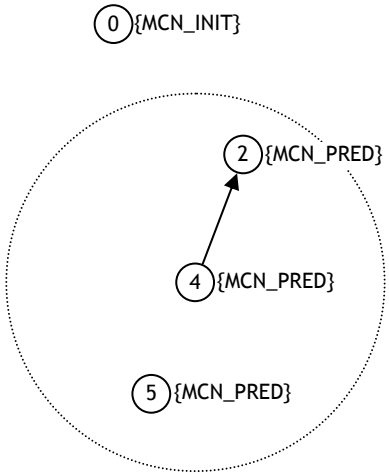
Following the path downstream towards the new member, any intermediate node about to take part in the packet forwarding process for the first time has to check for availability of three times the QoS bandwidth needed by the session, since it shares the bandwidth of the same neighbourhood with the two nodes immediately preceding and succeeding it. Once the multicast session starts, it receives packets from its predecessor, rebroadcasts them and allows its successor to forward the packets further downstream. For the general case, a forwarder needs $b_{s_k}$ to be available to its predecessor, requires $b_{s_k}$ for its own transmission and needs an additional $\phi_{i,s_k} b_{s_k}$ to be at the disposal of its successors.

In summary, nodes have to check for availability of necessary bandwidth according to their position within the multicast graph before accepting a request. For a member already forwarding packets of that session, this requirement is met automatically since the node has already been through this allocation process. Similarly, a receiver receiving a join

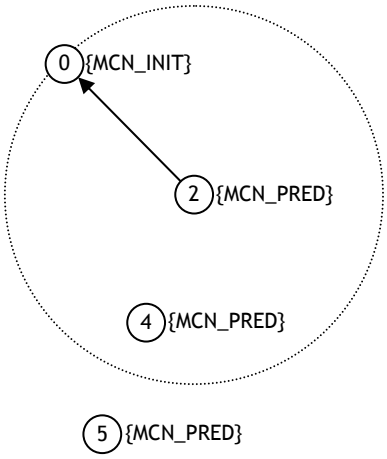
request does not present a problem to the system since it is aware of the resource allocation made previously by its immediate forwarder and also knows that it should check for availability of additional resources in order to reply the requests and become a forwarder.



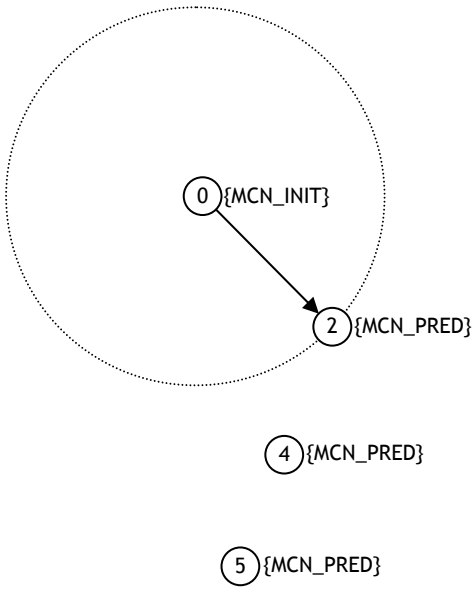

(a) $n_5$ checks its neighbourhood for $b_{s_0}$ and sends its join request




(c) $n_4$ checks its neighbourhood for $2\,b_{s_0}$ and forwards the request




(c) $n_2$ checks its neighbourhood for $3\,b_{s_0}$ and forwards the request




(d) $n_0$ checks its neighbourhood for $2\,b_{s_0}$ and sends its reply

Figure 4.3. The propagation of a join request along the virtual tunnel of bandwidth

The join process of AQM is explained in Section 3.4.1 with the help of an example depicted in Figure 3.15. Figure 4.3 shows the virtual tunnel of bandwidth approach on the same example, where nodes check for resource availability in a pipelined fashion. Node $n_0$ initiates a session with a bandwidth requirement of $b_{s_0}$. Having received the initiation message, $n_5$ wishes to join that session. (a) First, it checks if there is $b_{s_0}$ available in its neighbourhood to ensure a feasible request such that, once a path is found, a predecessor will be able to allocate that bandwidth to forward multicast data to it. Then it sends its request, which is forwarded further upstream by $n_4$ only if the latter also has enough free bandwidth in its neighbourhood. (b) Sharing the same neighbourhood with its predecessor $n_2$ and its successor $n_5$, $n_4$ will need two $b_{s_0}$ if it becomes a forwarder, $b_{s_0}$ for receiving multicast data from $n_2$ and another $b_{s_0}$ for forwarding it to $n_5$, which is just a receiver not forwarding any packets. (c) $n_2$, on the other hand, shares the resources of the same neighbourhood with $n_0$ and $n_4$. It checks for three $b_{s_0}$ since its immediate successor $n_4$ is not the requester and will also become a forwarder, which means that there will be a three-hop data flow in this neighbourhood. In other words, $n_2$ has to make sure that its successor $n_4$ can also forward the streaming data, after its predecessor $n_0$ and $n_2$ itself. (d) Finally, the initiator $n_0$ checks for two $b_{s_0}$ since it has to ensure that $n_2$ can forward the data packets it will send as a server. If enough bandwidth is available, $n_0$ sends a reply which is forwarded towards the requester following the same rules.

When it is time to allocate resources, each node only needs to reserve $b_{s_0}$, which is the amount of bandwidth required to forward the packets one hop downstream. The importance of the virtual tunnel approach is that it allows a sequence of three nodes to check whether they can support a session while sharing the same neighbourhood.

As more nodes join the multicast session and the multicast graph grows, the virtual tunnel of bandwidth approach converges to a general guideline which can be expressed as follows. For the session servers to sustain the continuous flow of the multicast data packets, the required bandwidth in the neighbourhood is the QoS bandwidth of the session multiplied by one plus the number of immediate neighbours which are also downstream forwarders of the session. This means that whenever the server sends a data packet, all of

its forwarders forward it one after the other. For the intermediate forwarders, the required bandwidth in the neighbourhood is one unit more than that of the servers, since they also have to receive data from their upstream forwarders. Finally, receivers only need an amount equal to the QoS bandwidth of the session that it actually used for their predecessors to forward multicast data to them.

### 4.3. Comments and Related Work

The estimation of the available bandwidth is an essential task for any QoS-related routing protocol. In this regard, the method for the calculation of the residual bandwidth, which is presented in this section, provides a sufficient estimate to the available bandwidth within the neighbourhood for a node. AQM nodes share this information among their neighbours with periodic greeting messages in order to keep themselves aware of the amount of bandwidth that they are allowed to allocate to new session requests.

On the other hand, the bandwidth estimation method described in this section is not a fundamental component of the AQM protocol. The available bandwidth calculated with this method is referred to as the effective bandwidth after having considered the channel contention overhead [92]. However, AQM is able to adapt any other technique for the prediction of the available bandwidth.

Available bandwidth estimation and monitoring is one of the essential tasks to accomplish for the development of an efficient methodology for bandwidth management [93]. There have been several proposals in the research literature for the estimation of the available bandwidth, where the wireless channel is generally described as a shared-access communication medium. The available bandwidth varies with the number of nodes contending for the channel and competition for bandwidth is not only end-to-end but also at every link [94].

Three methods are developed in [95] to predict the achievable bandwidth. According to the first method, each node broadcasts its own load information periodically to its one-hop neighbours, in addition to the load information of its two-hop neighbours. This way, each node gathers information on its three-hop neighbourhood and uses it for an

approximation of the achievable bandwidth. In the second method, the transmission delay is measured, which is inversely proportional to the service rate of the network, which is defined heuristically as the achievable bandwidth. Finally, the third method suggests that the nodes on a defined route also contend with each other and the achievable bandwidth is the minimum available on this route divided by the number of nodes on the route contending with the bottleneck node providing the minimum bandwidth.

The estimation of available bandwidth is considered the basis for admission control. In [96, 97], an admission control and dynamic bandwidth management scheme is proposed. The bandwidth requirement of an application is converted to a channel time requirement and weighted according to the requirements of other connections. The channel time is then shared between connections. The weights are dynamically adjusted as the available bandwidth changes. A central bandwidth manager obtains the bandwidth requirements from the connections at the beginning. It controls admission at connection establishment and redistributes bandwidth shares at connection teardown. It rejects the connection if the minimum channel time requirement cannot be supported.

Another computation method is developed by the ad hoc QoS on-demand routing (AQOR) algorithm in order to estimate the available bandwidth and perform accurate admission control [98]. Admission control decisions are made by every node based on the analysis of the traffic in the shared channel access network. To this end, each node sends hello packets to its neighbours, which contain information on self-traffic. The total traffic flow in the neighbourhood of a node is given as the sum of self-traffic and the traffic of the neighbours, which is deduced from the hello packets received. The available bandwidth is found by subtracting this value from the maximum transmission bandwidth.

As mentioned earlier in this section, these or other bandwidth estimation techniques can be combined with AQM. Depending on the network conditions and the support of the lower layers, important issues such as contention and interference can be taken into account. Thus, more sophisticated prediction schemes can be implemented by AQM in order to make the reservation decisions more accurate.

# 5. COMPUTATIONAL EXPERIMENTS

The evaluation of QoS multicast routing performance in ad hoc networks requires criteria that are both qualitative and measurable. The main concern of this section is to test the efficiency of AQM in providing multicast users with QoS and satisfying the service requirements of multimedia applications. Therefore, it is necessary to define the criteria to measure user satisfaction both at member and session levels, whereas previous research mainly focuses on quantitative aspects of efficiency such as packet loss ratio and control overhead. Thus, a new performance metric is introduced first, in addition to conventional criteria inspired by the related work in the literature. Next, the simulation settings are presented, under which the proposed multicast routing system is compared to a non-QoS protocol. The settings include the logical node structures, the usage scenarios, the network parameters and the mobility assumptions designed and combined in such a manner that a fair comparison can be made between AQM and its competitor in a realistic ad hoc network environment. Finally, simulation results are presented graphically and interpreted. The results show that, by applying novel QoS management techniques, AQM significantly improves multicast efficiency for members as well as sessions.

## 5.1. Performance Metrics

Distributed, loop-free, on-demand operation is a very important qualitative property for ad hoc networks [99]. However, the evaluation of ad hoc routing protocols also necessitates quantitative metrics, which can be measured to give a notion about their internal efficiency. The goodput, i.e., the packet loss rate defined by the ratio of the number of packets received to the number of packets transmitted, the average end-to-end delay and throughput as well as the control overhead are widely used to evaluate ad hoc routing protocols. These are also adopted by ad hoc QoS routing protocols [24-26], in addition to QoS-oriented metrics such as the success ratio defined by the number of accepted connections divided by the number of connection requests, the average path cost [27, 28] and the incompleteness ratio defined by the number of broken connections divided by the number of successful QoS requests [24, 27].

The efficiency of ad hoc multicast routing protocols is further measured by their data packet delivery ratio, data forwarding, i.e., packet replication efficiency defined by the number of data packets transmitted per original data packet and control overhead [40, 50, 52, 53, 70, 74]. Other metrics used are average delay, percentage of lost packets, number of control packets received by each node [63-65], multicast tree lifetime [79, 80] and energy consumption [83]. Finally, the multicast session success rate is defined by the number of accepted receivers divided by the number of requests to join a session [81, 82].

In the following sections, the performance criteria used in this thesis are presented. Divided into three groups, they evaluate AQM with regard to its efficiency as a QoS-aware multicast routing protocol in general, its performance in satisfying QoS requirements in particular and its consequences on the other aspects of the mobile ad hoc network.

### 5.1.1. Satisfaction of Session Members

The success of a QoS multicast routing system depends primarily on the satisfaction of its members. In this regard, the most important criterion for the QoS-related multicast routing decisions made by AQM is the improvement in the ratio of session members satisfied by the perceived quality of their applications. It is one of AQM's main concerns that network resources are not excessively utilized to avoid possible collisions, packet loss and delay due to overload and keep the QoS conditions at a satisfactory level. Once accepted to a session, the QoS status perceived by individual nodes during the course of the session is vital. Therefore, it is necessary to observe the changes in member-level QoS. The member QoS sustainability ratio $Q_{Member}$ is defined to evaluate this aspect of AQM and formulated as follows:

$$Q_{Member} = 1 - \frac{d}{a} \tag{5.1}$$

where $d$ is the number of members dropped off a session due to insufficient QoS and $a$ represents the number of nodes accepted to sessions as receivers. The decision on the sustainability of QoS is based on a combination of various other QoS metrics presented in Section 5.1.2. The equation gives the percentage of members which are served by the ad

hoc network with acceptable QoS during their entire session membership. The member QoS sustainability ratio is an important criterion for the evaluation of member satisfaction since it is also a measure of the percentage of members experiencing severe delay and loss problems due to allocations exceeding the resource limits of the network.

The success rate of member satisfaction is an important criterion for the performance of a multicast routing protocol providing QoS. However, it has to be taken into account that the member satisfaction achieved by the prevention of overload has an effect on the system, which can be observed by the percentage of users that are admitted to the multicast sessions. An efficient QoS multicast protocol should not allow its user admission rate to drop unacceptably as a result of the application of QoS restrictions. In other words, the majority of the users who wish to join a multicast session should still be admitted even with QoS limitations. Thus, the member acceptance ratio $A_{Member}$ is formulated as follows:

$$A_{Member} = \frac{a}{g} \tag{5.2}$$

where $g$ is the total number of join requests issued by all ad hoc nodes. Their ratio reflects the success rate of AQM in accepting a node's request to join a session. As mentioned above, the member acceptance ratio is an important performance metric which is used in previous research efforts [27, 28, 81, 82]. It is important that a QoS-aware multicast protocol can maintain a good balance between these two aspects of member satisfaction.

### 5.1.2. Individual QoS Criteria

There are several important metrics which give a particularly detailed insight on the performance of a QoS-aware routing protocol. Some of these metrics are the end-to-end delay, jitter, interarrival time and loss rate of data packets [99]. In fact, QoS can be defined as the differentiation of network services by these metrics. On the other hand, the data streaming capability is an essential feature of a multicast protocol that aims to provide multimedia services with QoS in an ad hoc network. Therefore, the packet interarrival time is considered particularly important for AQM, in addition to delay, jitter and loss. Thus, AQM is compared to its non-QoS competitor with regard to these criteria. In addition,

these metrics are used by AQM during the course of the sessions to observe the streaming of data and decide on the members' QoS sustainability, which is introduced in Section 5.1.1. Therefore, it is necessary to examine these criteria in more detail.

### 5.1.3. Effects on Network Load

It is inevitable that the computational overhead of a routing protocol increases with its complexity. However, it is possible to keep this overhead at an acceptable level while adding QoS functionality to a protocol, especially in order to deal with the effects of mobility, the changes in topology and the issues of scalability. Thus, the member control overhead $C_{Member}$ is formulated as follows:

$$C_{Member} = \frac{c}{z + f + a} \tag{5.3}$$

where $c$ represents the total number of multicast control packets received and processed by the nodes of the ad hoc network, $z$ is the total number of session servers and $f$ is the total number of forwarders. The sum of $z$, $f$ and $a$ gives the total number of active nodes in the network. An active node is a session member participating in at least one multicast session as a server, forwarder or receiver. Thus, the division gives the number of control packets per multicast member to manage and maintain the AQM system. As mentioned above, the member control overhead has also been used in previous research efforts [63-65].

Another important factor which deserves attention is the effect of AQM on the best-effort traffic performance. It is generally assumed that users may maintain background activities such as e-mail communication, Internet browsing or file transfer while attending interactive sessions. Therefore, AQM is also evaluated regarding the background traffic.

## 5.2. Simulation Settings

The simulations are conducted using OPNET Modeler 11.5 Educational Version and Wireless Module [100]. AQM nodes are modelled in three layers with application, session and network managers. The application manager is responsible for selecting the type of

application, setting its QoS requirements, as well as making decisions on session initiation, termination, join and leave. The session manager is responsible for declaring new sessions initiated by its application manager to other nodes, sending requests for sessions its application manager wishes to join, keeping lists of sessions, members and requests of other nodes, processing and forwarding their information messages and taking part in their join processes when necessary. The network manager is responsible for packet arrival and delivery, in addition to broadcasting periodic greeting messages and receiving other nodes' greeting messages in order to process them to derive free bandwidth information. A node can take part at only one application at a time as a server or receiver. However, it can participate in any number of sessions as a forwarder as long as QoS conditions allow. The usage scenarios consist of open-air occasions such as search and rescue efforts and visits to nature in an area with boundaries, where a network infrastructure is not available.

The non-QoS protocol developed for comparison purposes resembles basically a modified version of MAODV [40, 41], which is the tree-based multicast extension of the AODV protocol [42, 43]. MAODV utilizes the information collected during the unicast route discovery, which is not implemented in the non-QoS protocol developed for the simulations to achieve fair comparison conditions. MAODV maintains sequence numbers for multicast groups, which are updated by the group leaders, to ensure that the most recent route to the multicast group is used. Like AQM, the non-QoS protocol supports multiple sessions as well as multiple service classes simultaneously. However, it does not make any intelligent decisions based on QoS availability when responding to session join requests. In the non-QoS protocol, all sessions are announced along the network and all nodes can join all sessions regardless from bandwidth and delay limitations.

Simulations are repeated 10 times for each data point and results are aggregated with a 95 per cent confidence interval in a multicast scenario with a set of four QoS classes to represent various applications coexisting in the system. Nodes initiate or join sessions according to a certain probability. Generated sessions are assigned randomly according to their relative frequencies to one of the four QoS classes defined in Table 5.1. To comply with the sample bandwidth requirements and delay tolerance characteristics given as part of the QoS definitions, nodes are restricted to certain minimum bandwidth and maximum hop count regulations. In other words, a node is allowed to join a session only if it can find

a path to the server with more bandwidth available than the minimum amount and less hops away than the maximum allowed. In the simulations, a low delay tolerance corresponds to a maximum allowed hop distance of three, whereas a high delay tolerance is represented by a maximum of four hops. Apart from this, there are no limits to the size of the multicast groups.

Table 5.1. QoS requirements of application classes

| QoS Class | Application Type | Bandwidth Requirement | Average Duration | Delay Limit | Interarrival Time |
|---|---|---|---|---|---|
| 1 | Voice conversation | 128 Kbps | 300 s | 10 ms | 33 ms |
| 2 | Streaming music | 256 Kbps | 900 s | 50 ms | 33 ms |
| 3 | Video conference | 512 Kbps | 600 s | 10 ms | 40 ms |
| 4 | Streaming video | 2 Mbps | 1,200 s | 50 ms | 40 ms |

The effect of mobility on the performance of AQM is observed under the random waypoint mobility model [101, 102]. In contrast to previous performance evaluations in the research literature, which limit their simulations to a few minutes, one hour of network lifetime has been simulated to get a realistic impression of the aggregated behaviour of multiple multicast sessions being maintained simultaneously in a distributed manner. The parameters of the mobility model and other simulation settings are given in Table 5.2.

Three sets of simulations are conducted with these common parameters. The first set examines the effect of network density on AQM. In this set, the size of the physical coverage area of the network is the actual variable. However, the average number of neighbours within a node's transmission range is used in the figures in order to present the results with a clearer representation of network density. The value is the multiplication of the network population with the ratio of a node's transmission area to the whole network area. The former two are constants given in Table 5.2. The second set, whereby the percentage of the sessions which belong to the heaviest service class is the variable, aims to test the effect of the changes in multicast traffic load on AQM. In this set, the other classes share the remaining occurrence probability equally. Finally, the third set is

conducted to observe the effect of background traffic load on AQM. In this set, the file size of the background data, which is transferred between two neighbours, is the simulation variable. For each node, there is a time gap defined by the background inactivity period between the end of a file transfer and the start of a new request. The file transfer process is based on a modified version of the trivial file transfer protocol (TFTP) [103]. The results achieved by this set give an indication for AQM's reaction to changing background activity rate. Table 5.3 summarizes the variables used in the simulation sets.

Table 5.2. Simulation parameters for the performance evaluation

| Parameter Description | Value |
|---|---|
| Background inactivity period | 300 s (exponential) |
| Greeting message interval | 10 s |
| Maximum link bandwidth | 10 Mbps |
| Mobility model | Random waypoint |
| Node pause time | 10 – 40 s (uniform) |
| Node speed | 1 – 4 m/s (uniform) |
| Multicast inactivity period | 100 s (exponential) |
| Network population | 100 nodes |
| Session update message interval | 15 s |
| Wireless transmission range | 250 m |

Table 5.3. Simulation variables of the performance evaluation

| Set | Variable Description | Number of Nodes in Range | Heavy Multicast Class Ratio | Background Data File Size |
|---|---|---|---|---|
| 1 | Network density | 10 – 50 | 0.25 | 2 MB |
| 2 | Multicast traffic load | 20 | 0.20 – 1.00 | 2 MB |
| 3 | Background traffic load | 20 | 0.25 | 0 – 8 MB |

## 5.3. Performance Evaluation

Several important metrics are presented in Section 5.1 in order to evaluate the performance of AQM. Since a QoS-based protocol needs special criteria to show its quality, the QoS sustainability ratio is defined for session members by Equation 5.1. In addition, the member acceptance ratio defined by Equation 5.2 and the control overhead per member defined by Equation 5.3, which are inspired by previous research, are also used to show the price that mobile users have to pay for being able to run multimedia applications with certain QoS guarantees. Finally, conventional metrics such as end-to-end delay, interarrival time and loss rate for data packets are added to the performance criteria to observe the performance of AQM in more detail under these individual aspects of QoS.

Three simulation variables are selected to compare the efficiency of AQM to a non-QoS protocol under changing network conditions, namely the network density, the ratio of sessions that belong to the multicast class incurring heavy traffic and finally the size of the background data traffic. These are important criteria to judge a protocol's efficiency against the effect of the multicast activity rate as well as the multimedia and best-effort traffic load. In the following sections, AQM is compared to the non-QoS protocol described in Section 5.2 based on these criteria under the application scenarios given in Table 5.1.

### 5.3.1. Satisfaction of Session Members

As defined in Section 5.1.1, there are two performance metrics to evaluate the member satisfaction, which are the QoS sustainability and acceptance ratios of session members. These metrics should rather be interpreted together in order to see their relation. First, the effect of the changes in network density is observed on both metrics. Then, the effect of traffic load is evaluated by increasing the ratio of initiated sessions with higher QoS requirements in the network. Finally, the effect of background traffic on the performance is presented. A major conclusion drawn from the simulation results presented in this section is that there is a tradeoff between member acceptance and sustainability of QoS. Thus, AQM lets one of them degrade gracefully in order to maintain the other at an acceptable level when necessary. The logic behind these decisions is explained below.

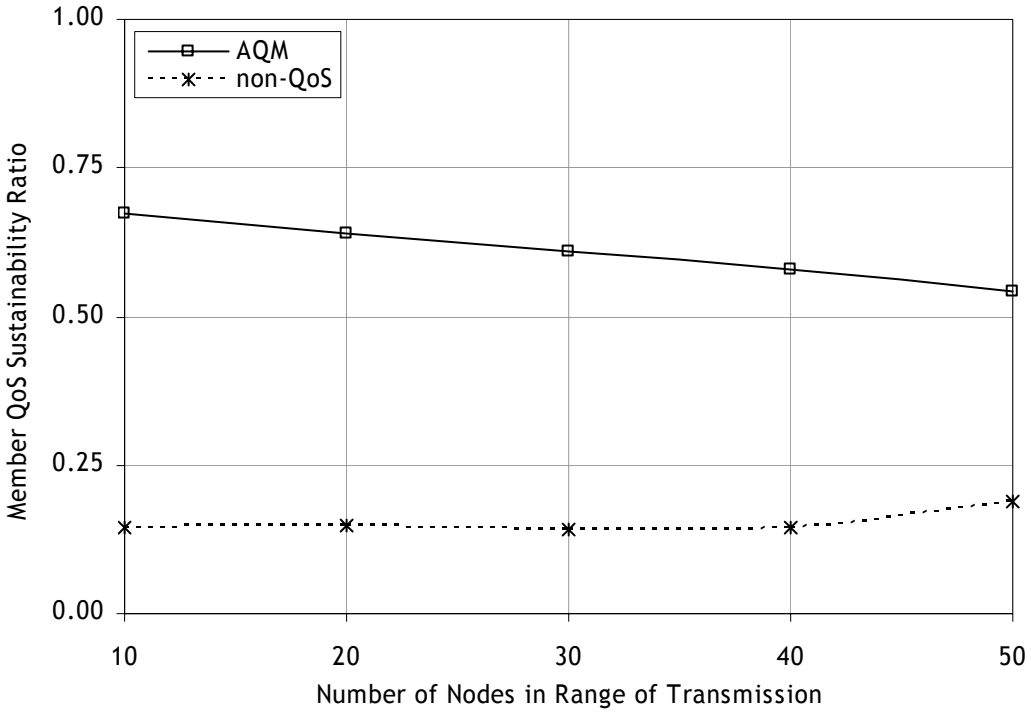



Figure 5.1. Member QoS sustainability ratio as a function of network density

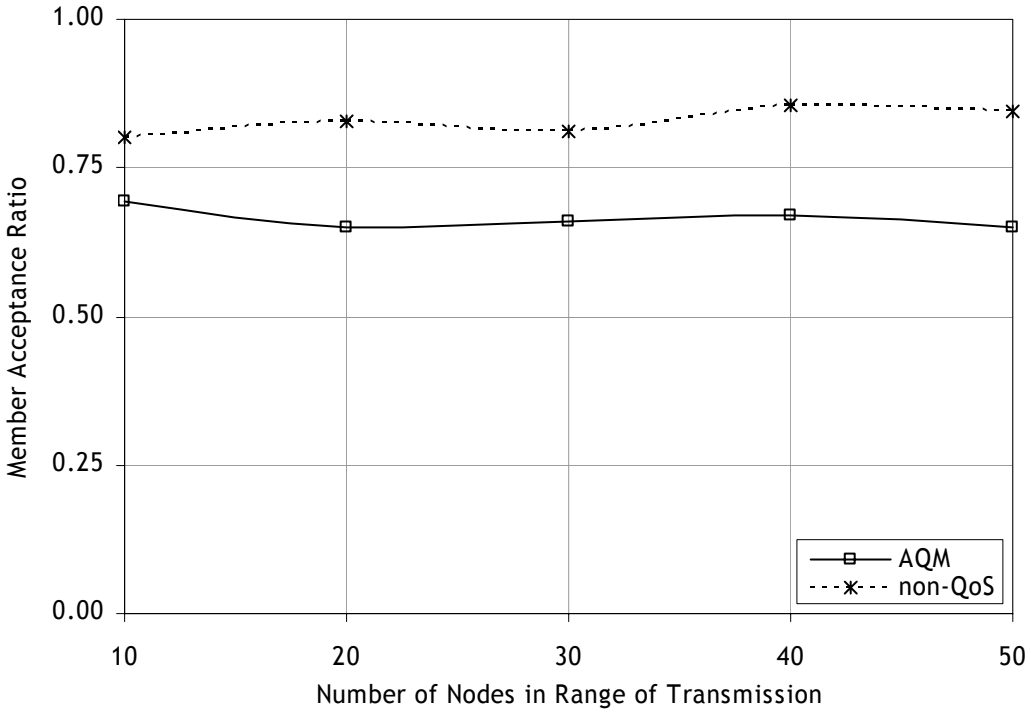



Figure 5.2. Member acceptance ratio as a function of network density

Figure 5.1 compares AQM to the non-QoS protocol regarding the ratio of accepted session members which can be provided with the required QoS for the duration of their membership. AQM monitors its receivers with regard to their perceived QoS such as their average data waiting times and packet loss rates for each of their session memberships. If a member experiences loss or delay beyond the limits of acceptable QoS, AQM decides that the QoS of the membership cannot be sustained any more and drops the member off the session to prevent further waste of resources. On the other hand, such a member is not dropped in the non-QoS protocol. In order to make a fair comparison, however, it is marked "should-have-been-dropped". It can be seen from the figure that AQM is able to sustain the membership QoS for a significant portion of the members once it accepts them to a session, whereas the non-QoS protocol can only provide poor QoS conditions to its users, mainly due to the fact that it accepts too many join requests without considering the resource limitations of the network. The member acceptance ratios of the protocols are displayed in Figure 5.2. As expected, AQM has a lower rate of member acceptance as a result of its stringent QoS restrictions and resource management precautions. However, it is still able to achieve an acceptance ratio close to its non-QoS competitor and preserve this ratio even as the network density increases and resources are shared by more nodes.

Nevertheless, AQM nodes experience a decrease in the ratio of satisfied members as the node density increases. The number of independent requests to be processed by several nodes simultaneously increases with network density. Several join processes start running in parallel and more than a feasible number of requests are accepted by their respective repliers before they can be informed on each other's allocations. Thus, the acceptance ratio is kept high as shown in Figure 5.1 at the cost of excessive resource allocation, which eventually leads to higher data packet loss rates due to increased interference and collisions. Consequently, AQM has to sacrifice an increasing portion of its session members as shown in Figure 5.2. It can be said that AQM is forced to adopt a reactive approach since it cannot predict an increase in simultaneous requests in the network. Improvements are possible for such cases with more frequent neighbour greetings and session updates as well as new join process facilities such as objection queries, which are introduced in Section 7, at the cost of increased control overhead. However, in comparison to the non-QoS protocol, AQM still provides QoS to a significantly higher ratio of its multicast session members with an acceptable ratio of rejected and dropped members.

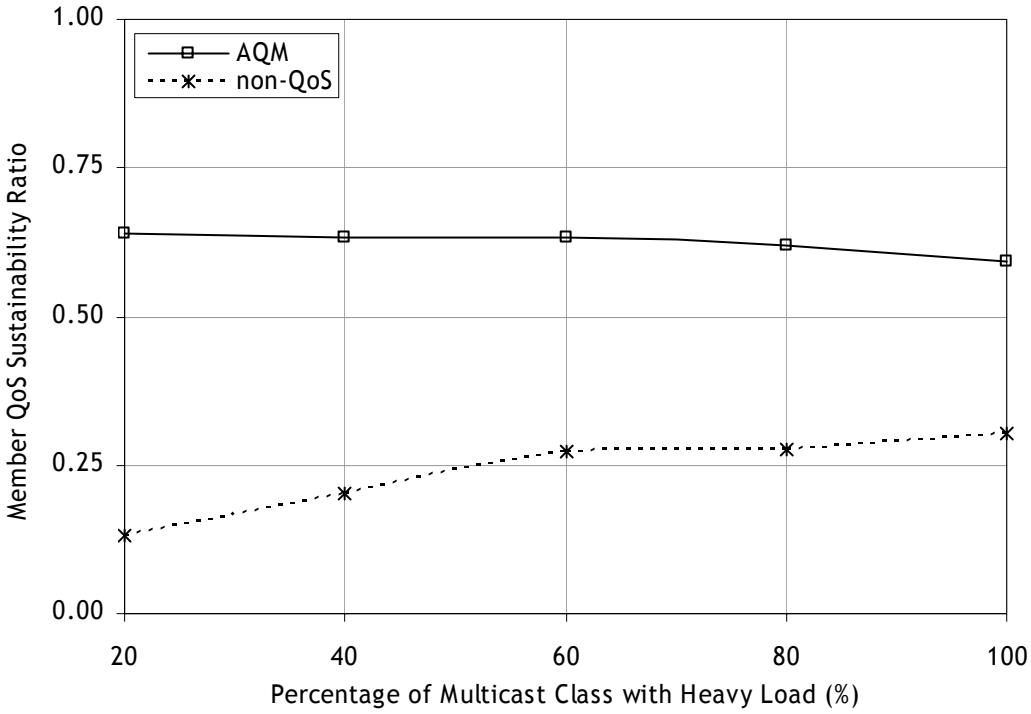



Figure 5.3. Member QoS sustainability ratio as a function of multicast traffic load

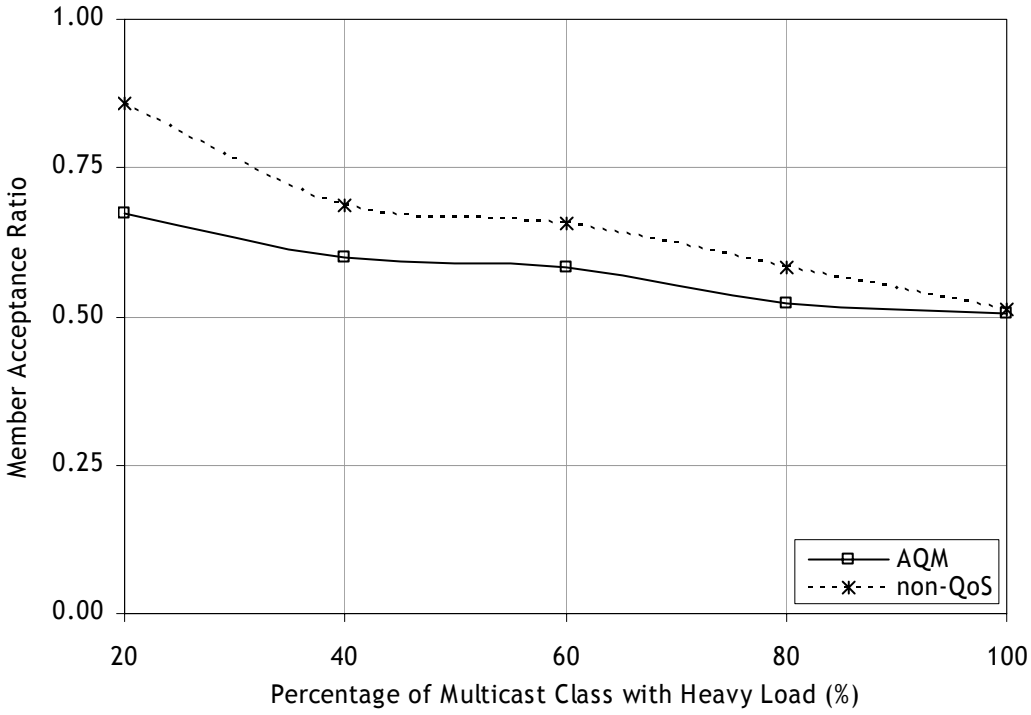



Figure 5.4. Member acceptance ratio as a function of multicast traffic load

Figure 5.3 and Figure 5.4 give the results of AQM performance with regard to QoS sustainability and member acceptance as the ratio of multicast sessions that belong to the class with higher QoS requirements increases. In contrast with the network density results presented in Figure 5.1 and Figure 5.2, AQM maintains the QoS level of its accepted members at the cost of decreasing its member acceptance ratio. The reason for this change in behaviour is the fact that AQM can use more up-to-date resource allocation information when there are fewer simultaneous join requests within a neighbourhood. In other words, AQM can act proactively by eliminating infeasible join requests as shown in Figure 5.4 and keep its member QoS sustainable as shown in Figure 5.3, if resource allocations at the end of simultaneous join processes are independent of each other. It can keep its nodes up-to-date regarding the QoS conditions in the network and the status of the existing sessions. AQM nodes do not accept new requests if they cannot afford the required bandwidth and hop count requirements. Thus, not all requests are granted an acceptance and the member acceptance ratio is generally lower than a non-QoS protocol. The increase in the QoS sustainability performance of the non-QoS protocol is the result of the decrease in its member acceptance ratio due to loss of control messages under heavy data traffic.

Another important aspect of the results presented in Figure 5.3 and Figure 5.4 is the fact that the non-QoS protocol cannot achieve the QoS sustainability rate of AQM even though its member acceptance ratio is very close to AQM for higher rates of heavy-class multicast traffic. This is a clear indication that AQM is more than admission control. A sustainable QoS rate close to that of AQM cannot be achieved merely by accepting join requests randomly at a rate close to that of AQM. AQM has many other features such as a QoS-controlled join process, resource allocation and hop count limitation which lead to a more balanced network load and increases the ratio of member satisfaction.

Figure 5.5 and Figure 5.6 show the QoS sustainability and member acceptance performances of AQM and its non-QoS competitor under the effect of increasing background traffic load. Since background traffic is defined as a best-effort service, it does not affect the decisions regarding the allocation of multicast resources. Therefore, the member acceptance ratios of both protocols remain relatively unchanged. When the file size becomes larger, on the other hand, it causes the probability of packet collisions to increase, which results in a slight degradation of member QoS sustainability.

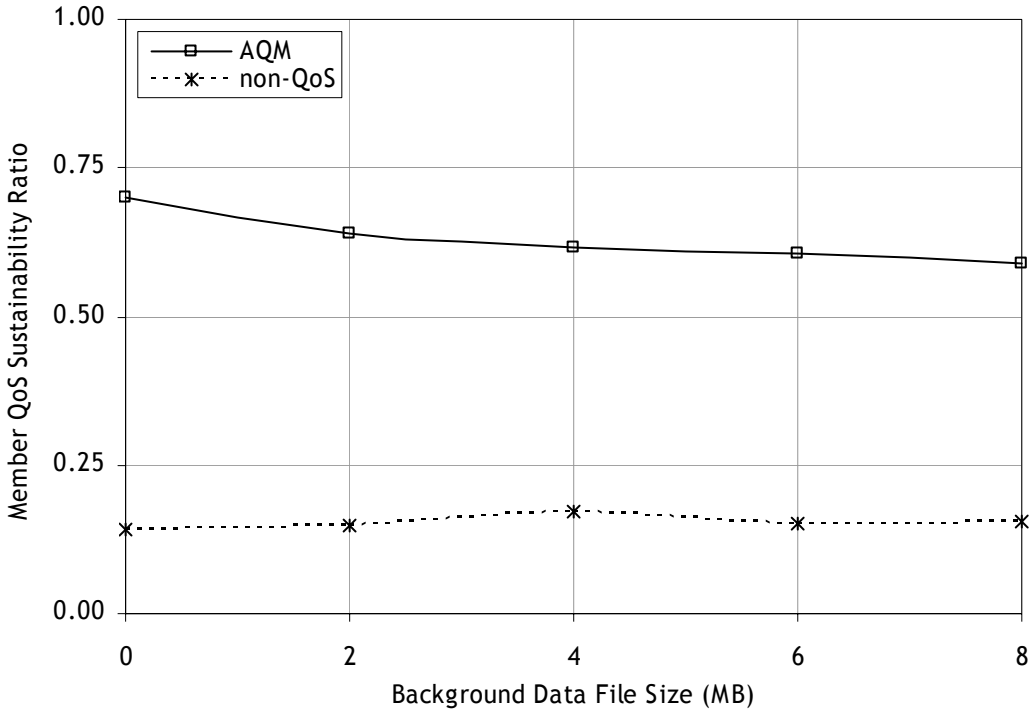



Figure 5.5. Member QoS sustainability ratio as a function of background traffic load

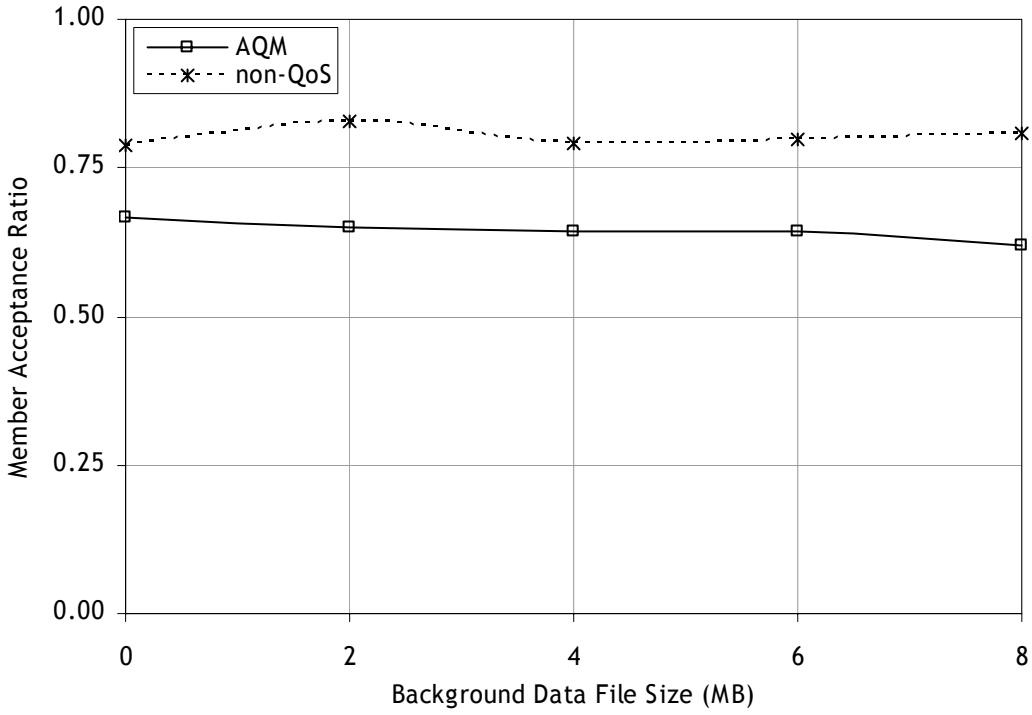



Figure 5.6. Member acceptance ratio as a function of background traffic load

It is also interesting to observe the background traffic success rate of AQM when the size of the transferred file increases, which is presented in Section 5.3.3.

The results presented in this section show that the QoS support provided by AQM significantly increases member satisfaction during multicast sessions. AQM outperforms the non-QoS protocol by sustaining QoS for a high ratio of members even under high traffic load conditions. It is a widely accepted assumption that dropped connections are generally more annoying than rejected ones. Thus, the member QoS sustainability ratio is a more important criterion than the member acceptance ratio, which means that AQM requires some improvements regarding its performance in dense networks. Moreover, an intermediate member with unacceptable QoS performance affects all its neighbours as well as all related sessions and is expected to cause collisions, packet losses and intolerable delays at the lower layers if it is not dropped. While the application of QoS restrictions causes more users to be rejected, the lack of these restrictions yields to performance degradation in the network. Without a policy to manage network resources effectively, users experience difficulties in getting any service as the bandwidth requirements increase.

### 5.3.2. Individual QoS Criteria

As mentioned in Section 5.1.1 and Section 5.1.2, AQM decides on the sustainability of a member's QoS based on a combination of various QoS metrics such as the ratio of lost data packets, the average of end-to-end delay and interarrival time. AQM continuously observes the data streaming performance of each session at each member and compares the results to the QoS requirements defined in Table 5.1. When a member experiences loss or delay exceeding the QoS limits for an unacceptable amount of time, AQM decides that the member should be forced to drop the session, since it is obvious that the member to be sacrificed and its potential successors cannot benefit from QoS-guaranteed service anymore. This is a necessary countermeasure in order to protect the rest of the network from performance degradation. Therefore, these three basic QoS metrics deserve a more thorough examination, which is provided in this section. Similar to the previous section, these individual performance criteria are evaluated with regard to three different variables. First, the effect of increasing the network density is observed. Then, the effect of multicast data load is evaluated. Finally, the effect of background traffic is presented.

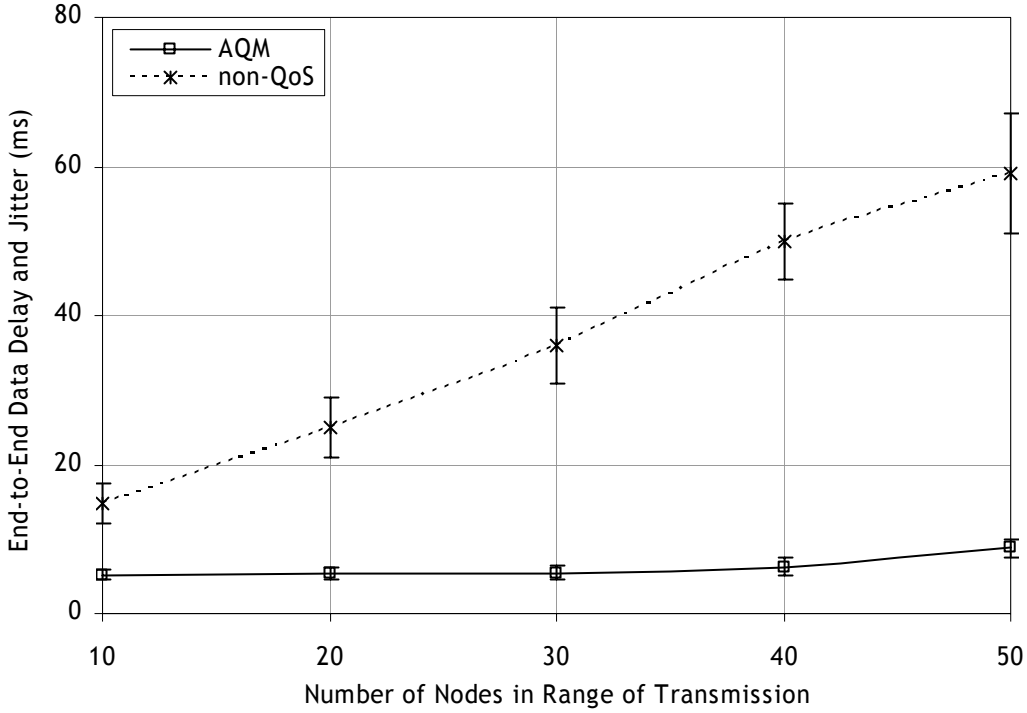



Figure 5.7. End-to-end data packet delay as a function of network density

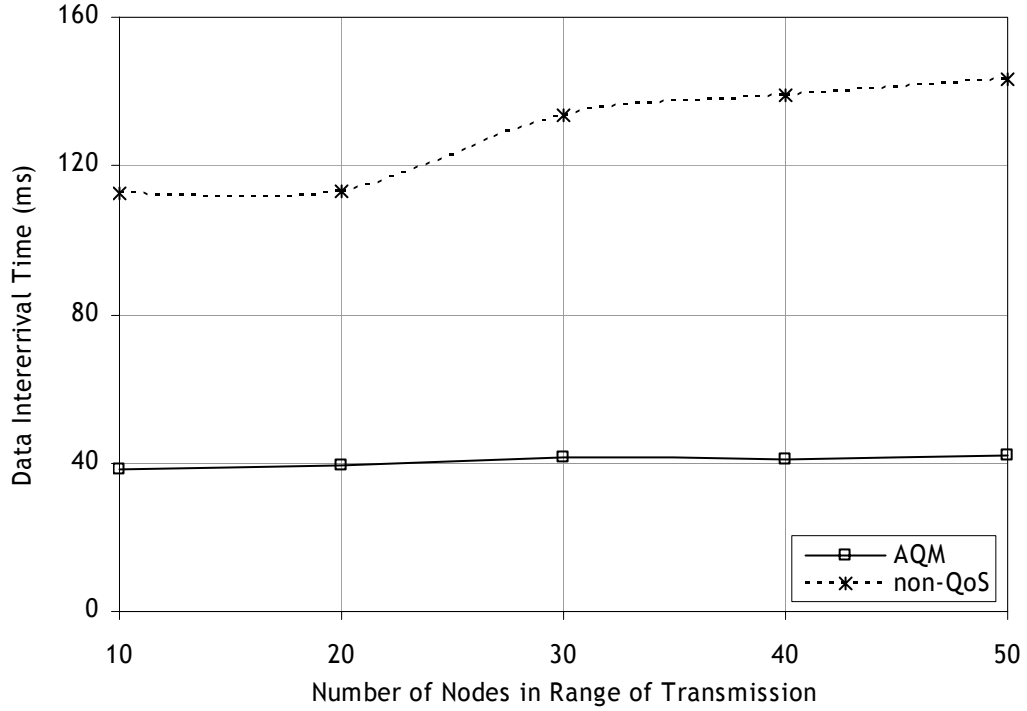



Figure 5.8. Data packet interarrival time as a function of network density

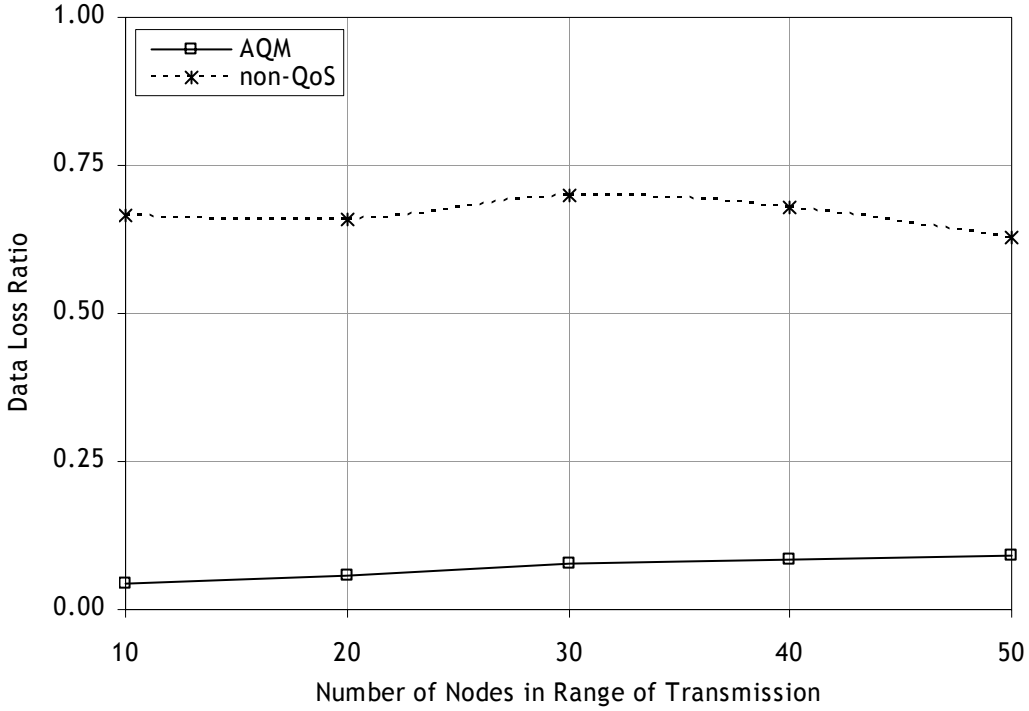



Figure 5.9. Data packet loss ratio as a function of network density

Figure 5.7 and Figure 5.8 display the average end-to-end delay and interarrival time of data packets, respectively, experienced in AQM and the non-QoS protocol. The reason for the end-to-end delay is contention, whereas the average interarrival time increases due to collisions. Both happen much rarer in AQM as a result of its ability to reserve resources and balance the network load. There is only a slight increase in the end-to-end delay of AQM as the network becomes denser and a similar behaviour is observed in its data interarrival time. AQM's delay variation is also much lower. The averages of the non-QoS protocol are higher than acceptable and increase drastically with network density. These results show that AQM can deliver data packets in a streaming fashion under relatively stable QoS conditions as required by multimedia applications. However, it should be noted that AQM achieves this performance at the expense of lower QoS sustainability.

Figure 5.9 compares the data loss rate of AQM to its non-QoS competitor. The dropping of members with unacceptable data streaming quality enables AQM to limit the network load in such a way that collisions are rare and data delivery rates are high for the remaining session members. Thus, the data loss rate increases only slightly with network density, which is an important achievement for a QoS-aware multicast routing protocol.

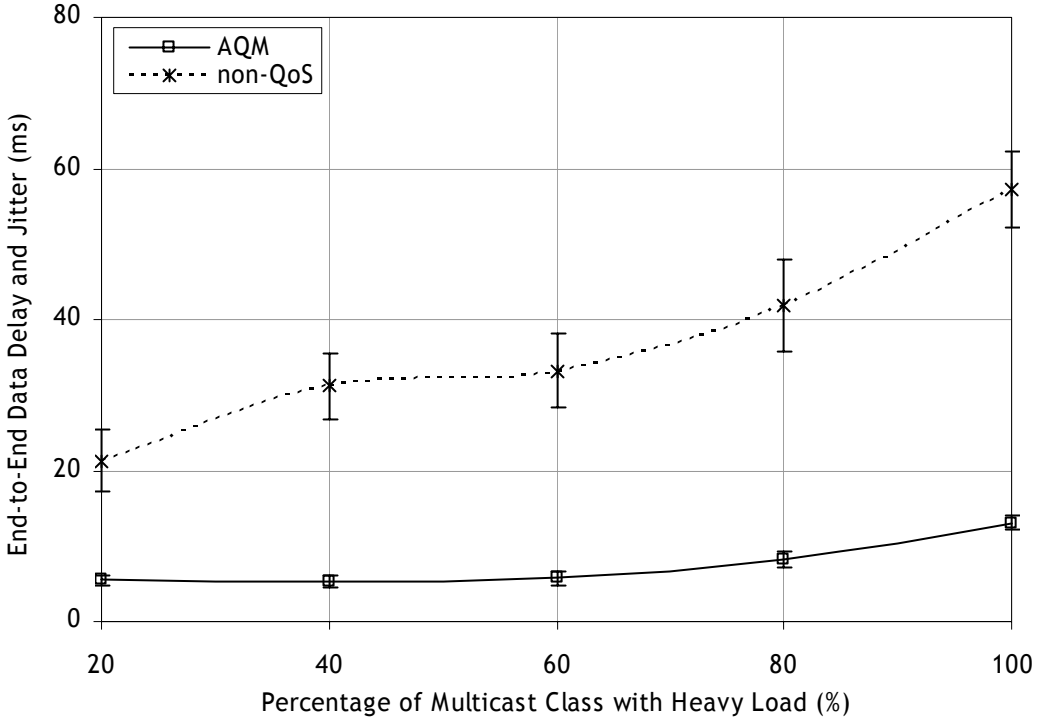



Figure 5.10. End-to-end data packet delay as a function of multicast traffic load

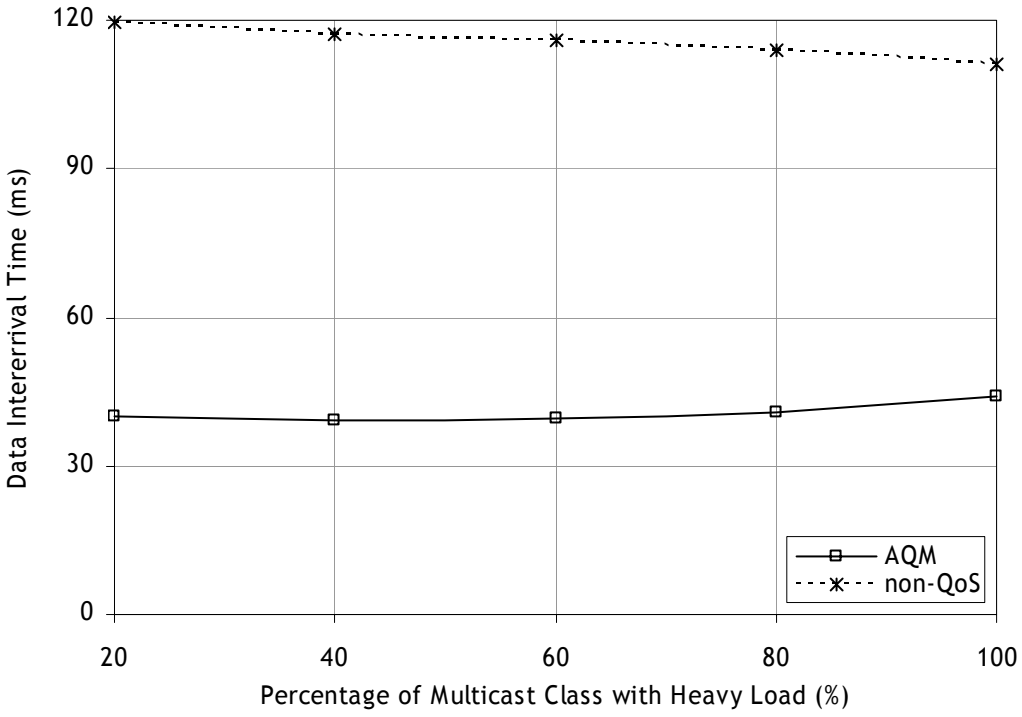



Figure 5.11. Data packet interarrival time as a function of multicast traffic load

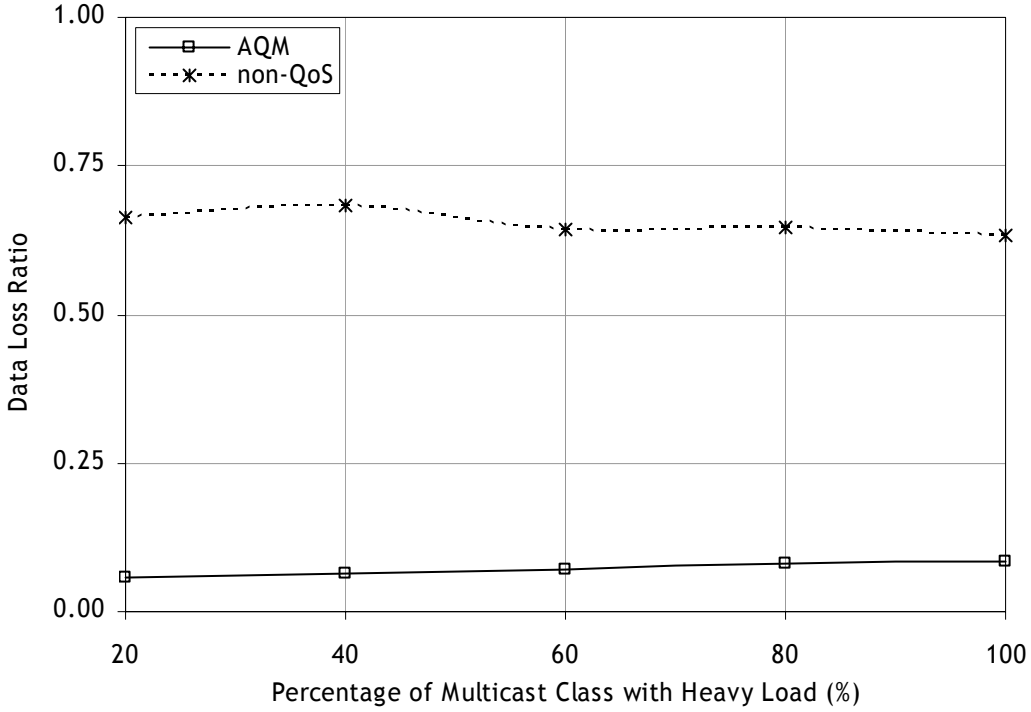



Figure 5.12. Data packet loss ratio as a function of multicast traffic load

Figure 5.10 shows the end-to-end delay and jitter values of AQM and its competitor. In both protocols, there is an increase in delay as the ratio of heavy-class applications increases. For AQM, this increase is a result of the larger transmission delays incurred by the large-size data packets. Since there are more heavy-class applications in the network, more large-size data packets are produced and more time is consumed to transmit them. However, the delay and jitter results are still within the QoS limits of the heavy-class application. On the other hand, the non-QoS protocol experiences additional delay due to contention at lower layers, which increases its end-to-end delay beyond acceptable limits.

Figure 5.11 and Figure 5.12 display the data interarrival times and loss rates of the protocols. Due to the larger data packets of the heavy class, the loss probability increases. Thus, AQM experiences slightly more data losses, which also affects the average time between data packet arrivals. Nevertheless, AQM is still able to keep these changes within allowed QoS limits by decreasing the member acceptance as necessary which is shown in Figure 5.4. The non-QoS protocol has also a lower acceptance ratio. Since the decrease is not based on network decisions but rather a result of lost control communication, however, it can only maintain a loss rate which is already too high for multimedia applications.

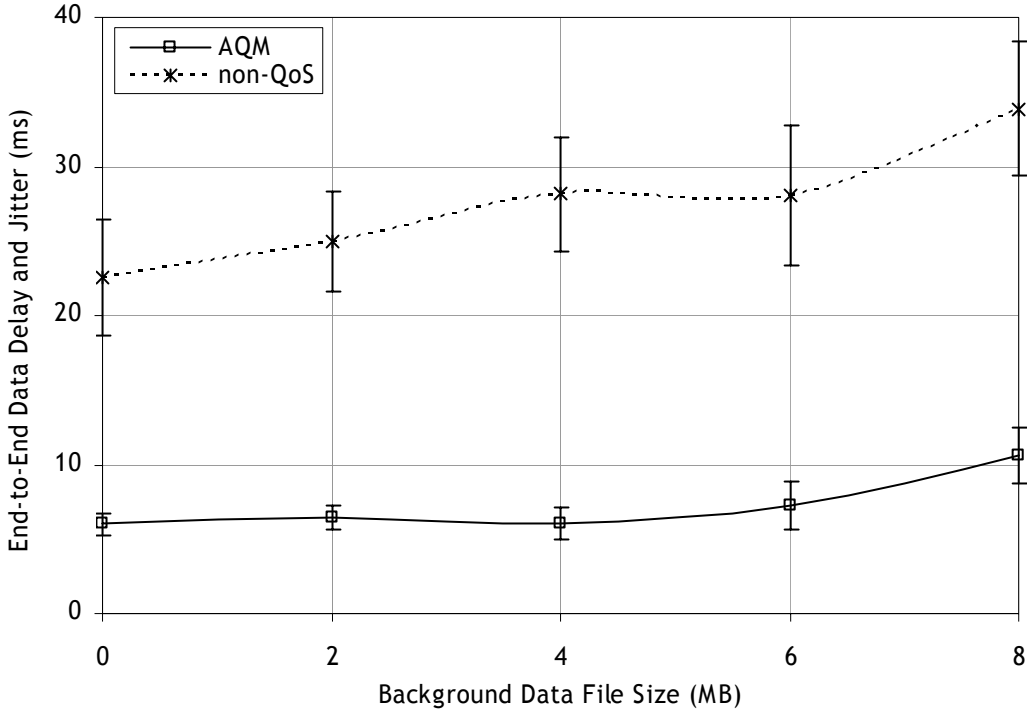



Figure 5.13. End-to-end data packet delay as a function of background traffic load

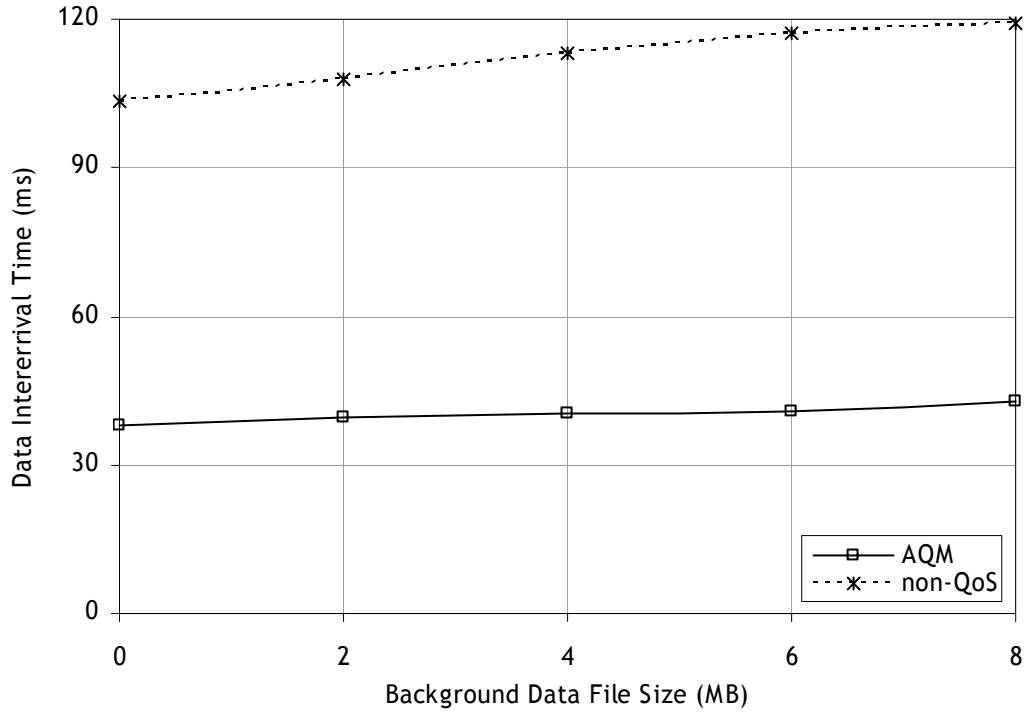



Figure 5.14. Data packet interarrival time as a function of background traffic load

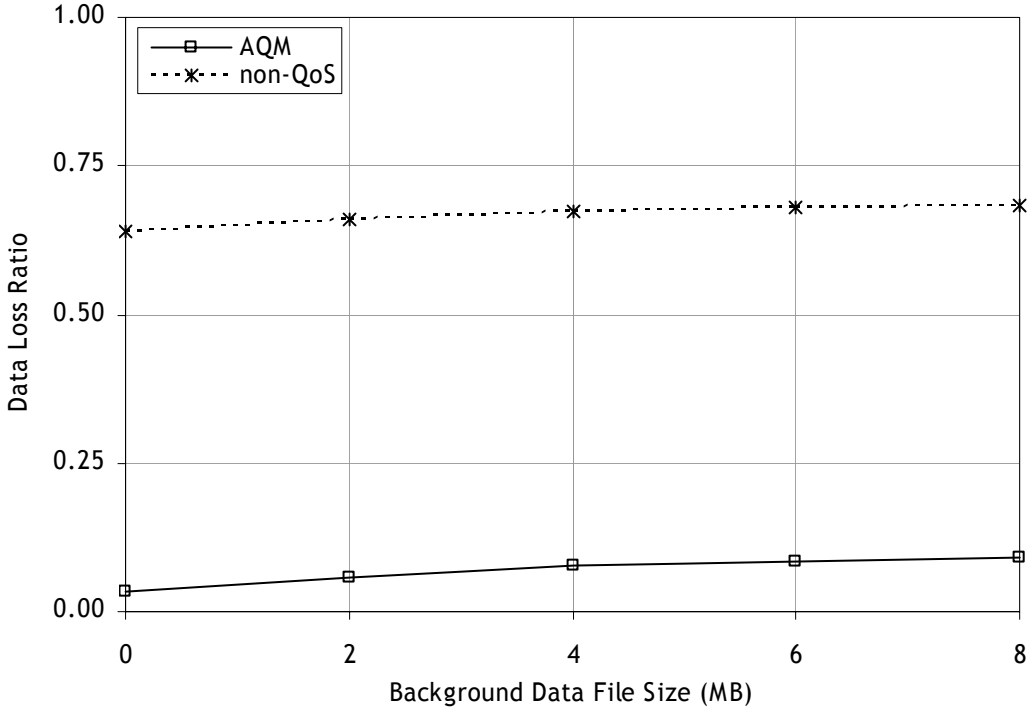



Figure 5.15. Data packet loss ratio as a function of background traffic load

The last group of results in this section examine the effect of background traffic load on the performance of AQM and its non-QoS competitor. As explained in Section 5.2, the background inactivity period is the gap between two consecutive background activities. Thus, a node requests a new file from another node, the size of which is the simulation variable, only after the inactivity period has passed after the completion of the last transfer.

Figure 5.13 shows how the background traffic affects the end-to-end delay and jitter experienced by the multicast traffic. Although the changes in delay and its variation are more severe for the non-QoS protocol than AQM, due to the fact that AQM experiences less contention, the results still show the need for a mechanism to differentiate between more and less delay-sensitive applications. On the other hand, Figure 5.14 and Figure 5.15 show that data interarrival times and loss rates of the multicast traffic are affected not that much by the background traffic. Since resources are reserved for a new member as soon as it joins a multicast session, fewer collisions are expected. However, the reservation of the resources does not include the scheduling of individual packets, which means that a multicast data packet may still have to wait for a background data packet to be transmitted. This topic is addressed and a solution for the problem is proposed in Section 8.

### 5.3.3. Effects on Network Load

AQM is a QoS-aware multicast routing protocol which is designed to operate in mobile ad hoc networks. Since there is a general trend in today's networks, which include conventional wired as well as wireless communications, towards the coexistence of QoS and best-effort traffic, it is imperative to examine the performance of AQM on best-effort traffic. In addition, the nature of ad hoc networks requires that such a protocol works with an additional overhead as little as possible. Therefore, it is also necessary that the control overhead incurred by AQM is evaluated to have an idea on its effects on the network load.

Figure 5.16 shows the performance of AQM and the non-QoS protocol with regard to their ability in maintaining a best-effort service in the background while also streaming data packets for multicast sessions as a multimedia service. The background traffic is implemented as a simple file transfer application running between neighbours. In order to achieve a fair comparison and evaluate AQM without the support of a transport layer, only the reliable unicast delivery feature of the underlying MAC layer is enabled, which is configured to retry sending an unacknowledged packet a predefined number of times and then proceed with the next packet. File transfers that can prevent their receivers from experiencing such losses or packet delivery timeouts are assumed to have completed successfully. Under these circumstances, AQM provides its users with a significantly better best-effort service than the non-QoS protocol. As the density of the network increases, the success rate of the background data delivery decreases for both. The reason for this decrease is the growing number of neighbours within a node's transmission range, which means that more nodes share the same amount of wireless resources and try to get multimedia as well as background service from each other. Thus, more interference and collisions occur and data is lost. Nevertheless, AQM succeeds in keeping its background success rate significantly higher than its competitor. In addition to taking the current bandwidth consumption of a node into consideration, AQM also uses the information on QoS requirements of the ongoing sessions within the neighbourhood of the node and adjusts the data rate of the background activity accordingly. The non-QoS protocol, on the other hand, has to adjust its best-effort data rate without this additional information and experiences more collisions due to its over-allocations. Thus, AQM's resource allocation strategies for multimedia sessions are also useful for background services.

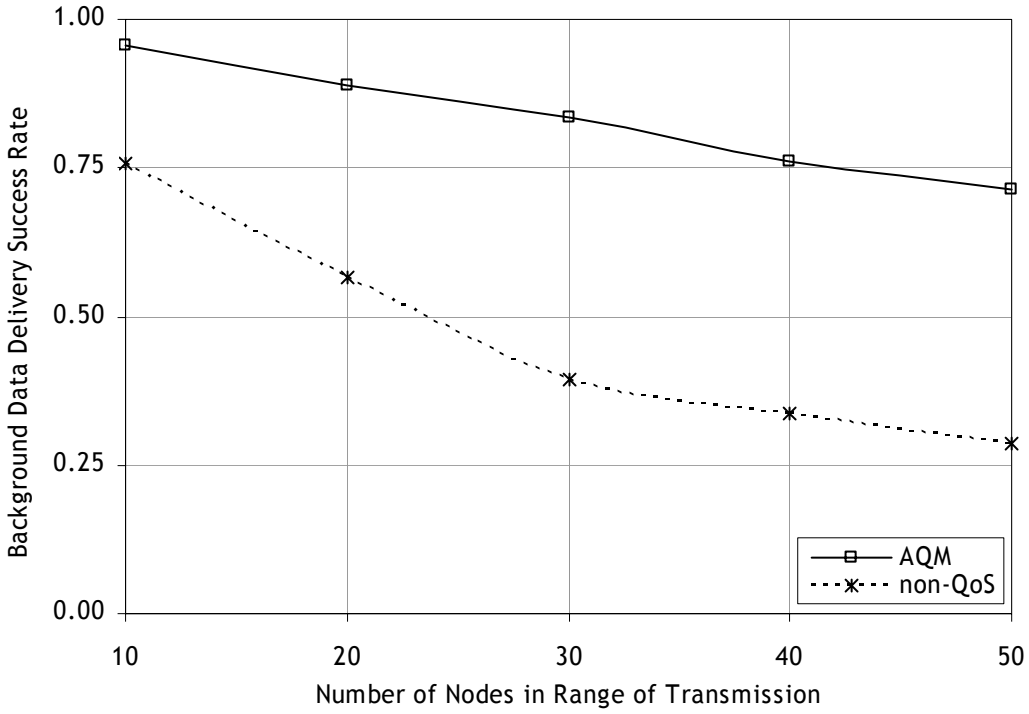



Figure 5.16. Background traffic efficiency as a function of network density

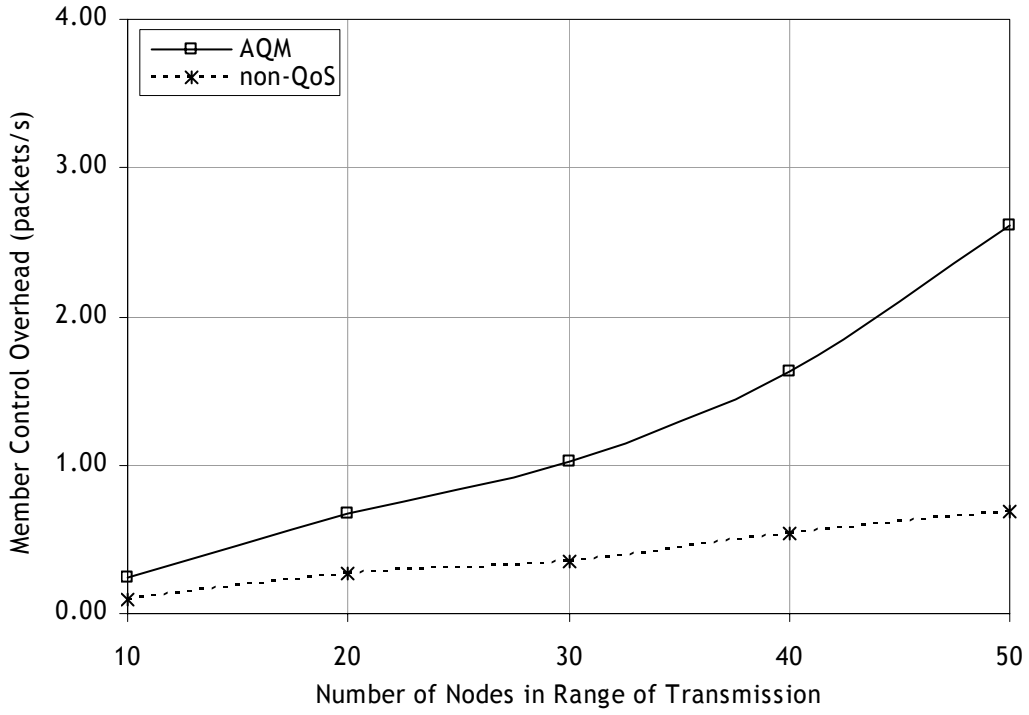



Figure 5.17. Member control overhead as a function of network density

Figure 5.17 compares the member control overhead of AQM to the non-QoS protocol as the network density increases. The unit of control overhead is defined as the number of packets received and processed by a session member per second. The average overhead increases slightly in a denser network, mainly due to more join requests and replies forwarded as a result of higher connectivity. Since the non-QoS protocol forwards these types of messages without QoS considerations anyway, the increased connectivity does not affect its control overhead as much as AQM. Another reason for AQM's increasing overhead is its member dropping process due to lack of acceptable QoS, which triggers subsequent actions at other session members both upstream as well as downstream. On the other hand, AQM eliminates infeasible join request at their sources and deals with less membership operations in general. Moreover, by rejecting some of the join requests, AQM cuts further communication with those nodes at an early stage of the process, whereas the non-QoS protocol communicates with all requesters until their routing information is delivered. Finally, AQM employs additional forwarders during the data streaming phase of its sessions, which increases the robustness of the multicast graphs and allows the protocol to experience fewer session losses to handle. These features help AQM prevent a much higher control overhead.

Figure 5.18 shows the background traffic efficiency of the protocols as the ratio of heavy-class applications increases. Similar to the results presented by Figure 5.16, background data success rates decrease as multicast traffic gets heavier, due to the fact that the wireless channel is occupied for longer periods of time in the attempt to transmit a higher ratio of larger multicast packets. Still, AQM is able to support a significantly higher portion of background data transfers by utilizing its resources efficiently whereas the non-QoS protocol becomes quickly overloaded as a result of excessive member acceptance.

Figure 5.19 evaluates the control overhead of AQM and its competitor with regard to multicast traffic load. The overhead of both protocols grows as the ratio of heavy class sessions is increased. The main reason for this behaviour common to both protocols is the fact that they reject more join requests, which yields to new requests and replies that are subsequently forwarded. AQM, on the other hand, facilitates additional control messages for status updates regarding its sessions and extra forwarders which are notified during multicast data flow to improve robustness.

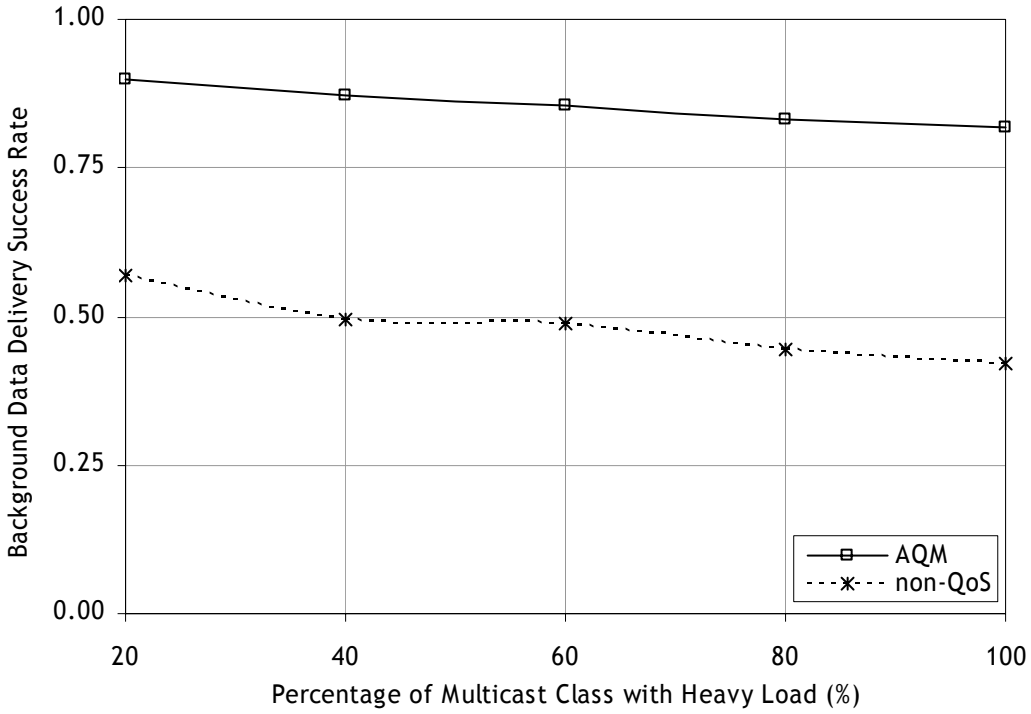



Figure 5.18. Background traffic efficiency as a function of multicast traffic load

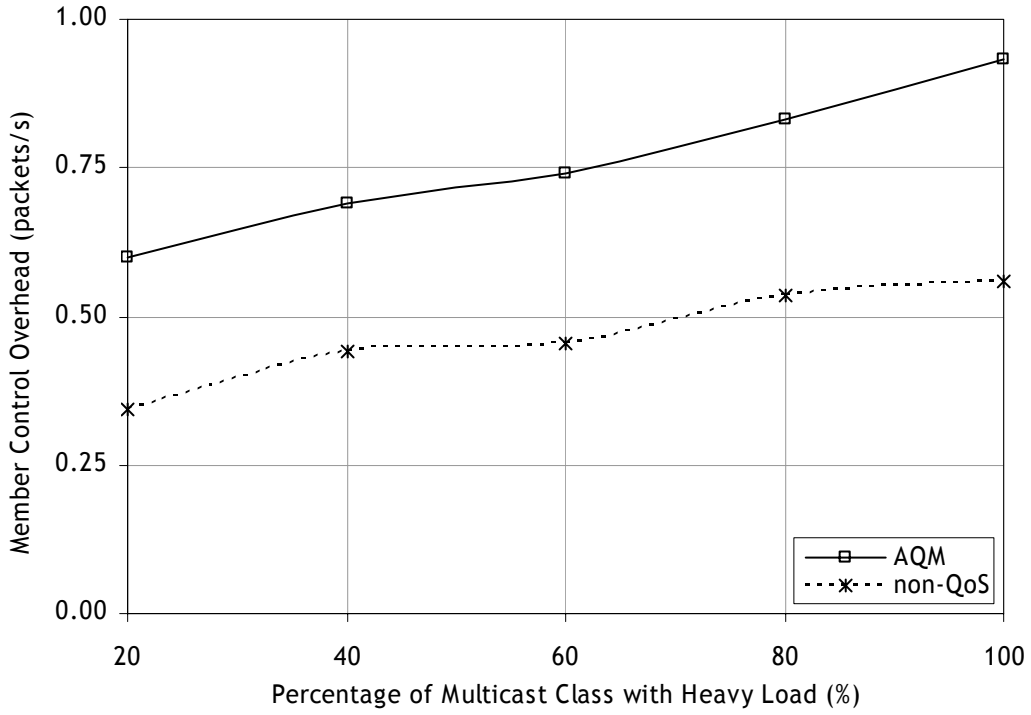



Figure 5.19. Member control overhead as a function of multicast traffic load

Figure 5.20 compares the background activity success rate of AQM to the non-QoS protocol as the file size of the background data grows. The success rate decreases, which is expected since the successful transfer of a larger number of consecutive packets is much harder in the dynamic neighbourhood of a mobile ad hoc network.

Finally, Figure 5.21 compares the member control overhead of AQM to the non-QoS protocol as the rate of background activity increases. The increase in the overhead is mainly caused by the loss of session update messages. Since these messages are periodic and rarer than other control messages as well as data packets, they are subject to a higher loss probability at times when they share the medium with background traffic. As a result of this, session losses occur slightly more frequently and the notification processes triggered by those incidents require more control messages to be sent.

Although the control overhead incurred by AQM is generally higher than the non-QoS protocol and the number of control packets per member grows significantly as the network density increases, it is worth mentioning that this overhead is actually very small when compared to the multimedia data traffic. Considering the average bandwidth requirements and session durations defined by the scenario in Table 5.1, the average data traffic per session is on the order of megabits per second. On the other hand, the size of the largest AQM control packet is around 200 bits, including various lower layer headers. AQM does not exchange the information on neighbours, members and sessions in the form of long lists. Therefore, it does not need to use variable-size control packets. The worst-case average control traffic per member is on the order of 0.5 kilobits per second, which is the rate experienced in a highly dense network. Thus, the increased overhead of AQM is still reasonable considering the fact that it achieves much higher member and session satisfaction for the users in the ad hoc network.

In this section, the control overhead experienced by the AQM nodes is observed in a general context, whereby the control packets are evaluated quantitatively without any classification. Thus, the aim of this section is to provide an overview of the control overhead incurred by AQM. In Section 6, the control overhead is analysed more thoroughly with particular emphasis on the session join process, which is the most interactive part of the protocol and therefore necessitates a deeper look.

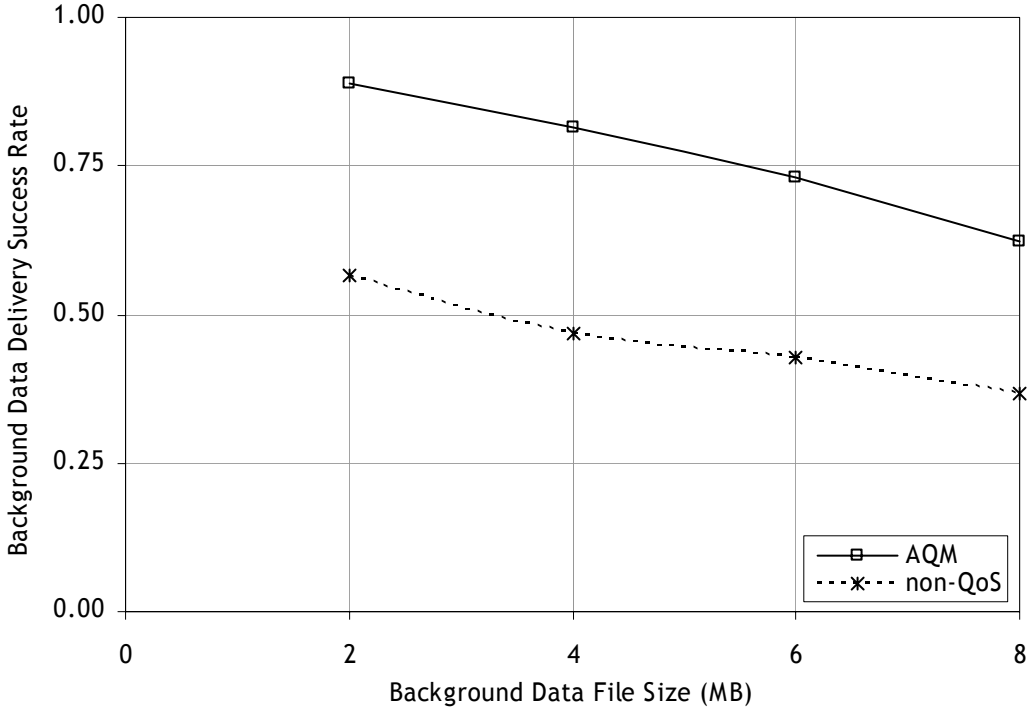



Figure 5.20. Background traffic efficiency as a function of background traffic load

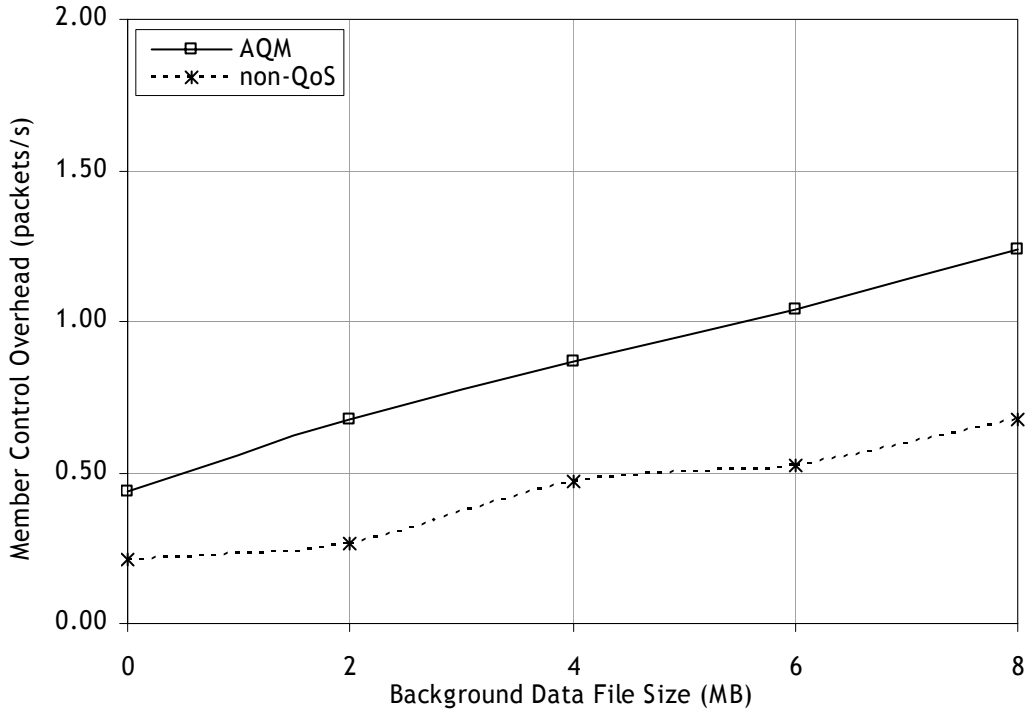



Figure 5.21. Member control overhead as a function of background traffic load

# 6. ANALYSIS OF THE CONTROL OVERHEAD

The control overhead incurred by AQM can be classified into three categories in accordance with the protocol structure. The session module sends session initiation and termination packets once per session, which are relatively rare events. In addition, there are the periodic session update packets. The overhead caused by these messages depends mainly on their frequency. Notifications for lost sessions are also sent by this module, which is expected to happen more frequently due to node mobility. However, assuming that there are much more members than sessions in the network at any instant, it can be argued that the control packets generated by the session module are only a fraction of the ones generated by the membership module. The greeting messages of the network module are another group of periodic messages, the use of which is explained in Section 3.5.1.

There are various packets sent by the membership module during the three-phase session joining process. The propagation of these packets, namely the request, reply and reserve messages, depends on the hop distance between the originator of the request and the nearest session member on the multicast tree, which can change with each request. It is therefore harder to perform an analysis of the control overhead incurred by the join requests. Since the join processes are expected to be the majority of all control events in the ad hoc network, however, they deserve this investigation. In the following sections, the join process of AQM and its impact on the system control overhead is analysed to provide an estimate of the amount of control messaging it incurs in the network.

There have been research efforts previously to analyse the expected number of hops for a source to reach a destination [104, 105]. The expected one-hop progress of a packet in the desired direction is defined as the distance between the sender and a receiver projected on a line connecting the source and the destination. It is formulated as a function of the number of neighbours, the node density, the transmission range and the distance from the source to the destination. The average number of hops between two nodes randomly placed within a circular area is then found by dividing their expected distance by the expected one-hop progress. However, the calculation of the average number of hops is not trivial and requires information on the Euclidian distance between nodes. Moreover,

the analysis is aimed at unicast communication. Therefore, a new approach is necessary, which is simpler and explicitly considers the properties of multicast communication.

In the following sections, the starting case for a session is considered as a first step, whereby the attempt of the first member to join the session is examined with regard to the possibilities of reaching the session server. Following the successful operation of the primary receiver to join the session, the possible behaviour of the subsequent join attempts is analysed. Finally, the three-phase join process is revisited for a single join attempt in isolation in order to observe the way a request propagates from its originator towards the session members and how it forces intermediate nodes to react to it. In the analysis, it is assumed that the protocol maintains multicast trees without evolving them to meshes, since this operation does not affect the results as explained at the end of Section 6.4.

## 6.1. Primary Receiver of a Session

Once a session is announced by its initiator, other nodes propagate the initiation messages throughout the network. As changes in the QoS conditions of the network are observed, session updates are utilized to inform the potential receivers of the availability of the session. Thus, any node that is aware of the session is eligible to join it.

At the beginning, the multicast tree consists of the server only, which is inactive since it has no receivers yet. Then, the tree starts growing due to the addition of receivers, which makes the finding of the tree easier for subsequent join candidates. Therefore, it is important to observe the events during the join process of the first candidate to the session.

**Definition 6.1:** Let $\nu$ denote *the number of nodes* in the ad hoc network, i.e., $\nu = |N|$. Let the network area be of circular shape with the radius *R*.

Assuming uniform node deployment, the node density $\rho$ can be found as follows:

$$\rho = \frac{\nu}{\pi R^2} \tag{6.1}$$

**Definition 6.2:** Let $R$ be an integral multiple of $r$, the node transmission range. Let $s$ be *the server of the session* located at the centre of the network area. Let $m_1$ be *the first receiver to join* the session.

The maximum number of hops $\gamma$ that $m_1$ needs to take in order to reach $s$ is:

$$\gamma = \frac{R}{r} \tag{6.2}$$

Figure 6.1 depicts the area that corresponds to the one-hop neighbourhood of $s$. It is possible to formulate the probability that $m_1$ is one hop away from $s$ as the ratio of the number of nodes within this area to the number of all the nodes in the network.

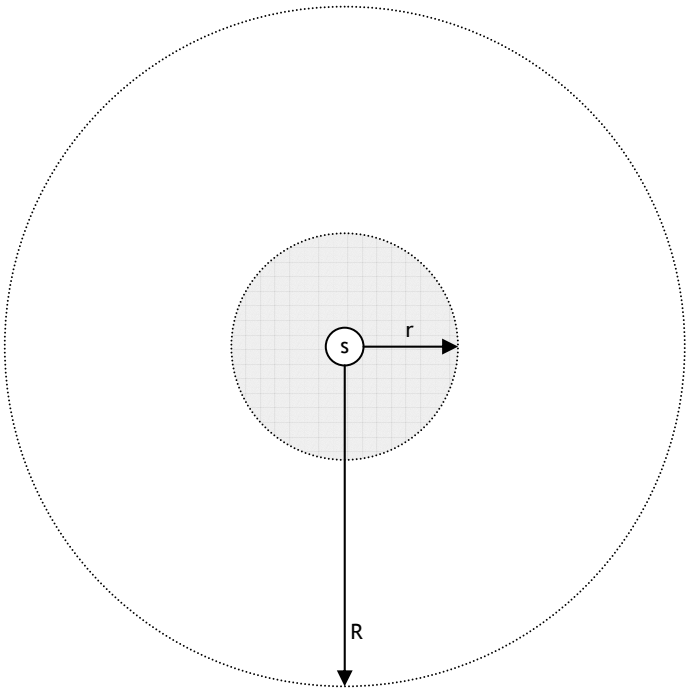



Figure 6.1. The one-hop neighbourhood of the session server

**Definition 6.3:** Let $H_1$ denote a discrete random variable having a probability mass function $p(h) = P\{H_1 = h\}$, which is defined as the probability that $m_1$ reaches $s$ in $h$ hops.

The probability that $m_1$ reaches $s$ in one hop is:

$$P\{H_1 = 1\} = \frac{\rho \pi r^2}{\nu} \tag{6.3}$$

$$P\{H_1 = 1\} = \frac{r^2}{R^2} \tag{6.4}$$

Equation 6.4 shows that the desired probability, which is the ratio of the number of nodes within the respective areas, is equal to the ratio of the areas of the respective circles, which, in turn, is equal to the ratio of the respective radii squared.

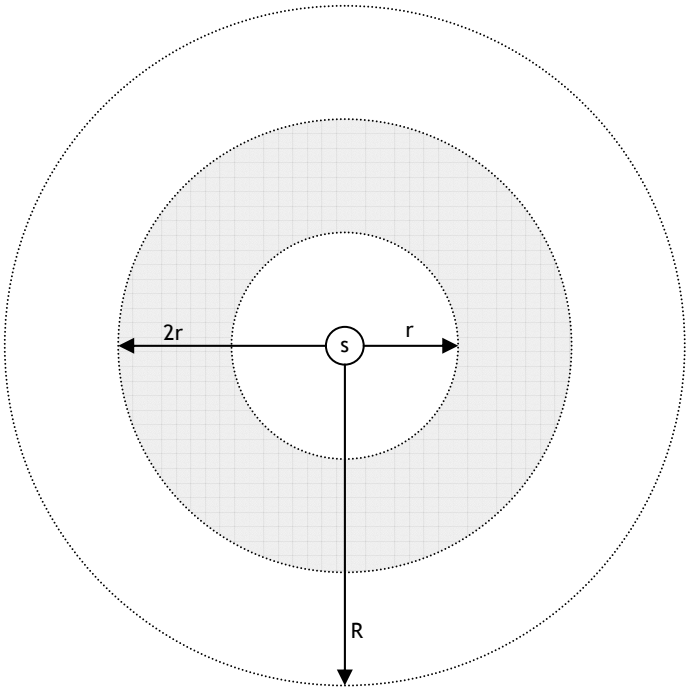



Figure 6.2. The two-hop neighbourhood of the session server

Figure 6.2 depicts the two-hop neighbourhood of *s*. Similar to the one-hop solution, the probability that the first candidate $m_1$ is exactly two hops away from *s* is equal to the ratio of the two-hop neighbourhood area of *s* to the area of the whole network.

$$P\{H_1 = 2\} = \frac{\pi\left[(2r)^2 - r^2\right]}{\pi R^2} \tag{6.5}$$

$$P\{H_1 = 2\} = \frac{3r^2}{R^2} \tag{6.6}$$

With the help of these results, the general case probability of $m_1$ being exactly in the $h^{th}$ neighbourhood of $s$, where $h \le \gamma$, can be formulated as follows:

$$P\{H_1 = h\} = \frac{(hr)^2 - [(h-1)r]^2}{R^2} \tag{6.7}$$

$$P\{H_1 = h\} = (2h-1)\frac{r^2}{R^2} \tag{6.8}$$

**Definition 6.4:** If $H_1$ is a discrete random variable having a probability mass function $p(h) = P\{H_1 = h\}$, then the expected value of $H_1$, which is the weighted average of the possible values of $H_1$, is defined as follows:

$$E[H_1] = \sum_{h:p(h)>0} h\,p(h)$$

The expected value of the number of hops which $m_1$ needs in order to reach $s$ is:

$$E[H_1] = \sum_{h=1}^{\gamma} h(2h-1)\frac{r^2}{R^2} \tag{6.9}$$

$$E[H_1] = \frac{r^2}{R^2}\left[2\sum_{h=1}^{\gamma} h^2 - \sum_{h=1}^{\gamma} h\right] \tag{6.10}$$

$$E[H_1] = \frac{r^2}{R^2}\left[2\frac{\gamma(\gamma+1)(2\gamma+1)}{6} - \frac{\gamma(\gamma+1)}{2}\right] \tag{6.11}$$

$$E[H_1] = \frac{1}{6\gamma}(\gamma+1)(4\gamma-1) \tag{6.12}$$

$$E[H_1] = \frac{2}{3}\gamma + \frac{1}{2} - \frac{1}{6}\gamma^{-1} \tag{6.13}$$

The last equation above shows that the expected number of hops for the first receiver $m_1$ to reach the server $s$ of the multicast session is mainly influenced by the ratio of the network radius $R$ to the transmission range $r$. The result is important since it directly affects the overhead incurred during the join process of a receiver, which is analysed in Section 6.3. It also has an impact on the overhead caused by subsequent join requests made for the same session, which is analysed in Section 6.2.

The relation between $E[H_1]$ and $\gamma$ can be further analysed by examining Equation 6.13 as follows:

$$\frac{E[H_1]}{\gamma} = \frac{2}{3} + \frac{1}{2}\gamma^{-1} - \frac{1}{6}\gamma^{-2} \tag{6.14}$$

As the physical size of the network area increases, while the transmission range remains constant, $\gamma$ approaches infinity according to Equation 6.2. Thus, the limit of the expected value of the hop count needed by the first receiver to reach the server is:

$$\lim_{\gamma \to \infty} \frac{E[H_1]}{\gamma} = \frac{2}{3} \tag{6.15}$$

$$E[H_1] = \frac{2}{3}\gamma \tag{6.16}$$

$$E[H_1] = \frac{2}{3}\frac{R}{r} \tag{6.17}$$

As mentioned previously, the expected number of hops grows linearly with the ratio between the physical size of the network area and the transmission range. Thus, an ordinary multicast routing protocol without any QoS constraints experiences increasing overhead as this ratio increases, regardless of the density of the nodes, which is not a

desirable property for the sake of scalability. On the other hand, the nodes that are far away from the server also suffer from high delays and packet losses, in addition to frequent disconnections, due to the length of the path between the server and themselves. Therefore, it is preferable that trees with high diameter values are avoided. This is why AQM applies hop count limitations as part of its QoS management strategies. The QoS class definitions followed by AQM practically restrict $\gamma$ in Equation 6.16 for each session separately, restricting the receivers to a virtual network border.

## 6.2. Subsequent Receivers

By joining the multicast session, the first receiver establishes a connection with the session server, which is the first path on the multicast tree with a number of intermediate nodes one less than the hop distance selected as the first set of forwarders. Based on the initial assumption of uniformly distributed nodes, the resulting multicast tree and its aggregated coverage area can be determined, which is the superposition of the transmission ranges of the current session members.

The fact that AQM favours paths with a minimum number of hops between the source and the destination necessitates that the sum of the Euclidian distances $d_{ij}$ between three consecutive nodes on the multicast tree has a lower bound, which is the transmission range $r$. If three nodes on the tree were placed closer than $r$, the first node would be able to by-pass the second one and reach the third node directly. Thus, the minimum distance between two nodes on the tree is the half of $r$ on the average. The maximum distance is the natural limit of the transmission range $r$.

**Definition 6.5:** Let $\bar{r}$ denote *the average distance between two consecutive nodes* on the multicast tree connecting $s$ and $m_1$.

Then $\bar{r}$ is bounded as follows:

$$\frac{1}{2}r < \bar{r} < r \tag{6.18}$$

**Definition 6.6:** Let $h_1$ be *the hop distance* between the first receiver $m_1$ and the server $s$, such that $1 \leq h_1 \leq \gamma$ .

An example is given in Figure 6.3, where $h_1$ equals four and the distance between the nodes is $r$. For the sake of simplicity, it is assumed that the nodes are located on a line. Thus, the coverage area to be examined is the upper limit with the maximum possible distance between two consecutive nodes.

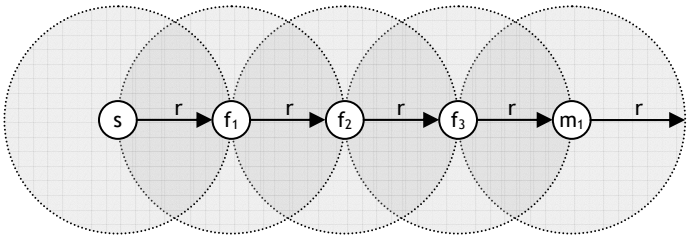



Figure 6.3. The multicast tree after the first receiver joins the session

Similar to the probability calculations made for the first receiver, the probability that the second receiver joins the multicast tree in one hop can be defined as the ratio of this area to the total area of the network. The above example is redrawn in Figure 6.4 to show the intersecting areas to be calculated with their $x$ values along the axis.

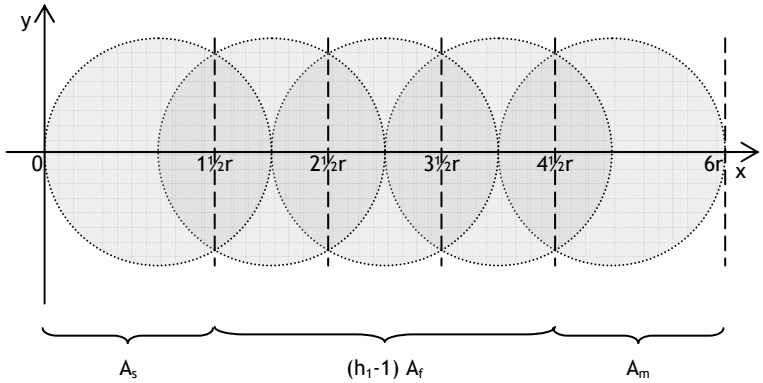



Figure 6.4. The coverage area of the multicast tree after the first receiver

Considering the general case, the number of the intermediate nodes is one less than $h_1$, which is equal to the number of the intermediate areas. Thus, the coverage area of the multicast tree after the first receiver denoted by $A_{T,1}$ is the superposition of the areas covered by the session server, the intermediate nodes and the receiver:

$$A_{T,1} = A_s + (h_1 - 1)A_f + A_m \tag{6.19}$$

**Definition 6.7:** Let $M = \{ s, m_l \}$, *the set of actual session members* including the server and the receivers. Let $F = \{f_h : 1 \le h \le h_1 - 1\}$, *the set of data forwarding nodes* in the session. Let $T = M \cup F$ , *the set of the nodes on the multicast tree*.

It can be seen from Figure 6.3 and Figure 6.4 that each node at the centre of its respective circle has a distance of *r* to the next one, which means that the centres also have a distance of *r* to each other. Thus, the general equation for these circles is formulated as:

$$(x - hr)^2 + y^2 = r^2 \tag{6.20}$$

where $1 \le h \le h_1+1$.

In order to formulate the integrals and compute the areas shown in Figure 6.4, the integration boundaries have to be calculated by eliminating *y* and solving the equations of two consecutive circles for $x_h$ as follows:

$$r^2 - (x_h - (h+1)r)^2 = r^2 - (x_h - hr)^2 \tag{6.21}$$

$$x_h = \frac{(h+1)^2 - h^2}{2} r \tag{6.22}$$

$$x_h = \left(h + \frac{1}{2}\right) r \tag{6.23}$$

where $1 \le h \le h_1$.

Solving Equation 6.23 for the various values of *h*, the $x_h$ values of all intersections are determined. Thus, the partial area around *s* to be included to $A_{T,1}$ as $A_s$ is the integral of the circle function given in Equation 6.20 over the interval $[0, x_1]$:

$$A_s = 2\int_0^{\frac{3}{2}r} \sqrt{r^2 - (x-r)^2}\, dx \tag{6.24}$$

where $h = 1$.

Similarly, the partial area around a forwarding node, $A_f$, can be formulated as the integral of the same function over the interval $[h-1, h]$:

$$A_f = 2\int_{\left(h-\frac{1}{2}\right)r}^{\left(h+\frac{1}{2}\right)r} \sqrt{r^2 - (x-hr)^2}\, dx \tag{6.25}$$

where $2 \leq h \leq h_1$.

Since the circles are identical, the calculation of the individual forwarding areas $A_f$ can be made easier by using the same integral as $A_s$ with shifted intervals. Thus:

$$A_f = 2\int_{\frac{1}{2}r}^{\frac{3}{2}r} \sqrt{r^2 - (x-r)^2}\, dx \tag{6.26}$$

$$A_f = 4\int_{\frac{1}{2}r}^{r} \sqrt{r^2 - (x-r)^2}\, dx \tag{6.27}$$

Finally, for the case where $h = h_1$, the area $A_m$ is identical to the area $A_s$. Thus, the formulation of the total area $A_{T,1}$ given in Equation 6.19 can be simplified as follows:

$$A_{T,1} = 2A_s + (h_1 - 1)A_f \tag{6.28}$$

Using Equation 6.24 and Equation 6.25 in Equation 6.19, the integral can be solved as follows:

$$A_{T,1} = 4\left\{\int_{0}^{\frac{3}{2}r}\sqrt{r^2-(x-r)^2}\,dx + (h_1-1)\int_{\frac{1}{2}r}^{r}\sqrt{r^2-(x-r)^2}\,dx\right\} \tag{6.29}$$

$$A_{T,1} = 4r\left\{\int_{0}^{\frac{3}{2}r}\sqrt{1-\left(\frac{x-r}{r}\right)^2}\,dx + (h_1-1)\int_{\frac{1}{2}r}^{r}\sqrt{1-\left(\frac{x-r}{r}\right)^2}\,dx\right\} \tag{6.30}$$

Substituting $\frac{x-r}{r} = \sin t$ and $\frac{1}{r}dx = \cos t\ dt$,

$$A_{T,1} = 4r^2\left\{\int_{-\frac{\pi}{2}}^{\frac{\pi}{6}}\cos^2 t\ dt + (h_1-1)\int_{-\frac{\pi}{6}}^{0}\cos^2 t\ dt\right\} \tag{6.31}$$

$$A_{T,1} = r^2\left\{h_1\left(\frac{\pi}{3}+\frac{\sqrt{3}}{2}\right)+\pi\right\} \tag{6.32}$$

The probability that the second receiver is within this area, which is also the probability that it can reach the multicast tree in one hop, is:

$$P\{H_2 = 1\} = \frac{A_{T,1}}{\pi R^2} \tag{6.33}$$

$$P\{H_2 = 1\} = \frac{r^2}{R^2}\left\{1+h_1\left(\frac{1}{3}+\frac{\sqrt{3}}{2\pi}\right)\right\} \tag{6.34}$$

However, it should be noted that Equation 6.33 only holds as long as $A_{T,1}$ is not larger than the network area. Otherwise, $P\{H_2 = 1\}$ is equal to one.

Using Equation 6.4, the relation of this result to the one-hop probability of the first receiver can be shown as:

$$P\{H_2 = 1\} = P\{H_1 = 1\} + \frac{r^2}{R^2} h_1 \left( \frac{1}{3} + \frac{\sqrt{3}}{2\pi} \right) \tag{6.35}$$

A second special case should be considered separately, where $\gamma$ is equal to one. In this case, $P\{H_1 = 1\}$ as well as $P\{H_2 = 1\}$ are equal to one since the transmission ranges cover the whole network. As a result of this, $P\{H_l = 1\}$ equals one for all receivers $m_l$.

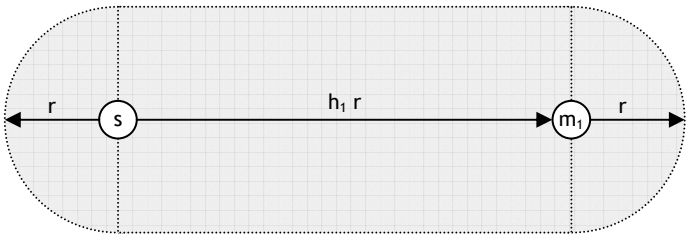



Figure 6.5. An approximation to the one-hop neighbourhood of the multicast tree

Although the coverage area of the multicast tree following the joining of the first receiver is computed exactly as given by Equation 6.32, an approximation is provided to simplify subsequent calculations, which is an upper bound to $A_{T,1}$. Thus, the area can be approximated as shown in Figure 6.5 and formulates as follows:

$$A'_{T,1} = r^2 (\pi + 2\, h_1) \tag{6.36}$$

The upper bound of the probability that the second receiver is in this area becomes:

$$P'\{H_2 = 1\} = \frac{r^2 (\pi + 2\, h_1)}{\pi R^2} \tag{6.37}$$

$$P'\{H_2 = 1\} = \frac{r^2}{R^2} \left( 1 + \frac{2\, h_1}{\pi} \right) \tag{6.38}$$

The results for $P\{H_2 = 1\}$ and $P'\{H_2 = 1\}$ show that the probability of reaching the multicast session in one hop increases for the second receiver when compared to the first.

$$P'\{H_2 = 1\} = P\{H_1 = 1\} + \frac{r^2}{R^2}\frac{2\,h_1}{\pi} \tag{6.39}$$

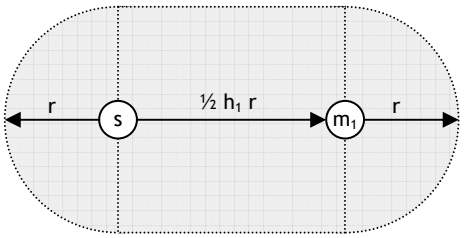



Figure 6.6. A lower-bound to the one-hop neighbourhood of the multicast tree

It should also be noted that this result provides a looser upper bound to $P\{H_2 = 1\}$ since the distance between the multicast nodes are assumed to be *r*, which is the maximum value possible. Considering the minimum distance between the nodes as defined in Equation 6.18, the lower bound of the coverage area is as approximated in Figure 6.6. It can be seen that only the area covered by the intermediate nodes shrinks and the lower bound of the probability to join the tree in one hop is formulated as follows:

$$P''\{H_2 = 1\} = P\{H_1 = 1\} + \frac{r^2}{R^2}\frac{h_1}{\pi} \tag{6.40}$$

Having found the coverage area of the one-hop neighbourhood of the multicast tree, the same approximation can be used to find its two-hop neighbourhood, which is illustrated in Figure 6.7. These approximations help the derivation of the probabilities $P'\{H_2 = h\}$ for the general case.

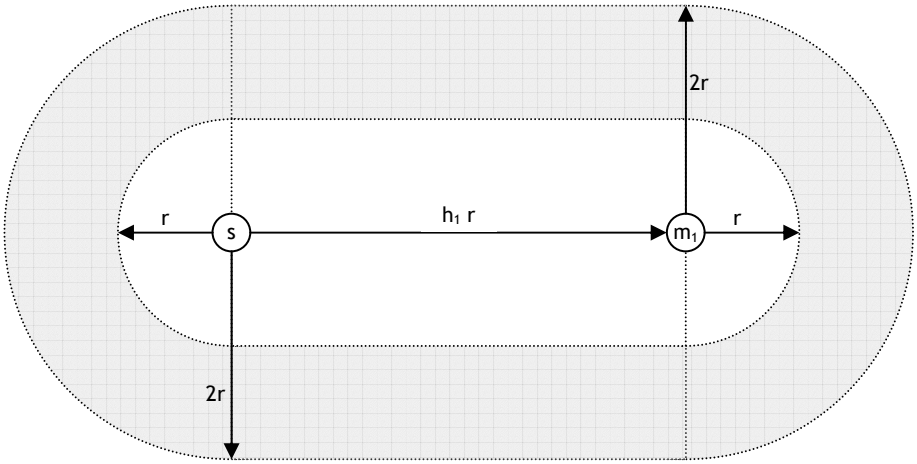



Figure 6.7. An approximation to the two-hop neighbourhood of the multicast tree

The area of the shaded region shown in Figure 6.7 is:

$$A'_{T,2} = 2r^2(2\pi + 2h_1) - A'_{T,1} \tag{6.41}$$

Hence, the area of the *h*-hop neighbourhood can be defined as follows:

$$A'_{T,h} = hr^2(h\pi + 2h_1) - A'_{T,h-1} \tag{6.42}$$

Thus, the probability for the second receiver to join the multicast tree in *h* hops is:

$$P'\{H_2 = h\} = \frac{A'_{T,h}}{\pi R^2} \tag{6.43}$$

$$P'\{H_2 = h\} = \frac{r^2}{R^2}\left[(2h-1) + \frac{2h_1}{\pi}\right] \tag{6.44}$$

Similar to Equation 6.33, Equation 6.43 only holds as long as $A'_{T,h}$ does not exceed the network area. On the other hand, there is a maximum value that *h* can take such that $A'_{T,h}$ is not larger than the network, since $A'_{T,h}$ is a function of both *h* and $h_1$, whereas *h* and $h_1$ are limited by Equation 6.2 and Definition 6.6, respectively.

Thus, the *h*-hop join probabilities for the second session member are:

$$P'\{H_2 = h\} = P\{H_1 = h\} + \frac{r^2}{R^2}\frac{2h_1}{\pi} \tag{6.45}$$

Similar to the results presented in Equation 6.39, this result provides an upper bound to $P\{H_2 = h\}$ since the distance between the multicast nodes are assumed to be *r* again. Considering the lower bound of the distance between the nodes, the lower bound of the probability to join the tree in *h* hops is formulated as follows:

$$P''\{H_2 = h\} = P\{H_1 = h\} + \frac{r^2}{R^2}\frac{h_1}{\pi} \tag{6.46}$$

The results for the *h*-hop join probabilities of the second receiver can be generalized for the subsequent members of the session. Since the coverage area of the multicast tree is an increasing function of the number of current receivers, any new member makes it easier for the next join request to reach the multicast tree in fewer hops. In other words, the relation between the *h*-hop join probabilities of $m_l$, any of the *l* receivers of the multicast session, can be formulated as follows:

$$P\{H_l = h\} \geq P\{H_{l-1} = h\} \geq \ldots \geq P\{H_2 = h\} \geq P\{H_1 = h\} \tag{6.47}$$

Using this relation, the *h*-hop join probabilities of the subsequent receivers can be approximated by $P\{H_2 = h\}$.

### 6.3. Overhead of a Join Process

The preceding sections analyse the behaviour of the first and second receivers in a session. Moreover, with the aid of some simplifications it is also possible to approximate the behaviour of the subsequent receivers. Given these approximations for the hop count of a new receiver to join the multicast tree, it is possible to compute the overhead incurred, if the number of the nodes involved in the process at each hop can be determined.

When a session is initiated, it is announced by the server throughout the network. As a result of the nature of the wireless medium, the session initiation messages propagate in the form of an expanding ring. Each node that is informed of the session for the first time forwards these packets only once, which guarantees the downstream flow of the information. The announcement is refreshed periodically by session update packets, which propagate following the same rules. Thus, the expanding ring structure, which groups the nodes according to their distance to the server in terms of hop count, remains intact throughout the session.

A join request propagates pretty much in the same fashion as a session initiation, in the form of an expanding ring centred at the node which originates the request. By definitions of AQM, only those nodes which are aware of the session, can satisfy its QoS requirements and are in a ring which is closer to the session initiator than the requester take the message into consideration. These nodes forward it further upstream towards the session server.

The ratio of these nodes in the network can be computed by finding the size of the intersecting areas of the two expanding rings that belong to the requesting and the serving nodes. An example case is illustrated in Figure 6.8, where the first requesting node $m_1$ is just outside the four-hop neighbourhood, or the packet propagation wave $w_{s,4}$ as it is labelled in the figure, of the server $s$. In this case, the number of nodes that become involved in the request-reply-reserve process can be found by calculating the sum of the areas $A_1$, $A_2$, $A_3$, $A_4$ and multiplying it with the node density, which is the division of the total number of nodes by the whole network area as formulated in Equation 6.1.

Following the example of Figure 6.8, the node $m_1$, which has a distance slightly more than four $r$ to the server $s$, finds itself in $w_{s,5}$. Thus, the join request has to propagate five hops, at the last of which it arrives at the server. This means that there are four groups of nodes between the requester and the server, which forward the request upstream. The first group to process the request consists of those nodes that are within the intersection of $w_{m,1}$ and $w_{s,4}$. These nodes are one-hop closer to the server than the requester. The second, third and fourth groups involved are formed similarly. They are shown in Figure 6.9.

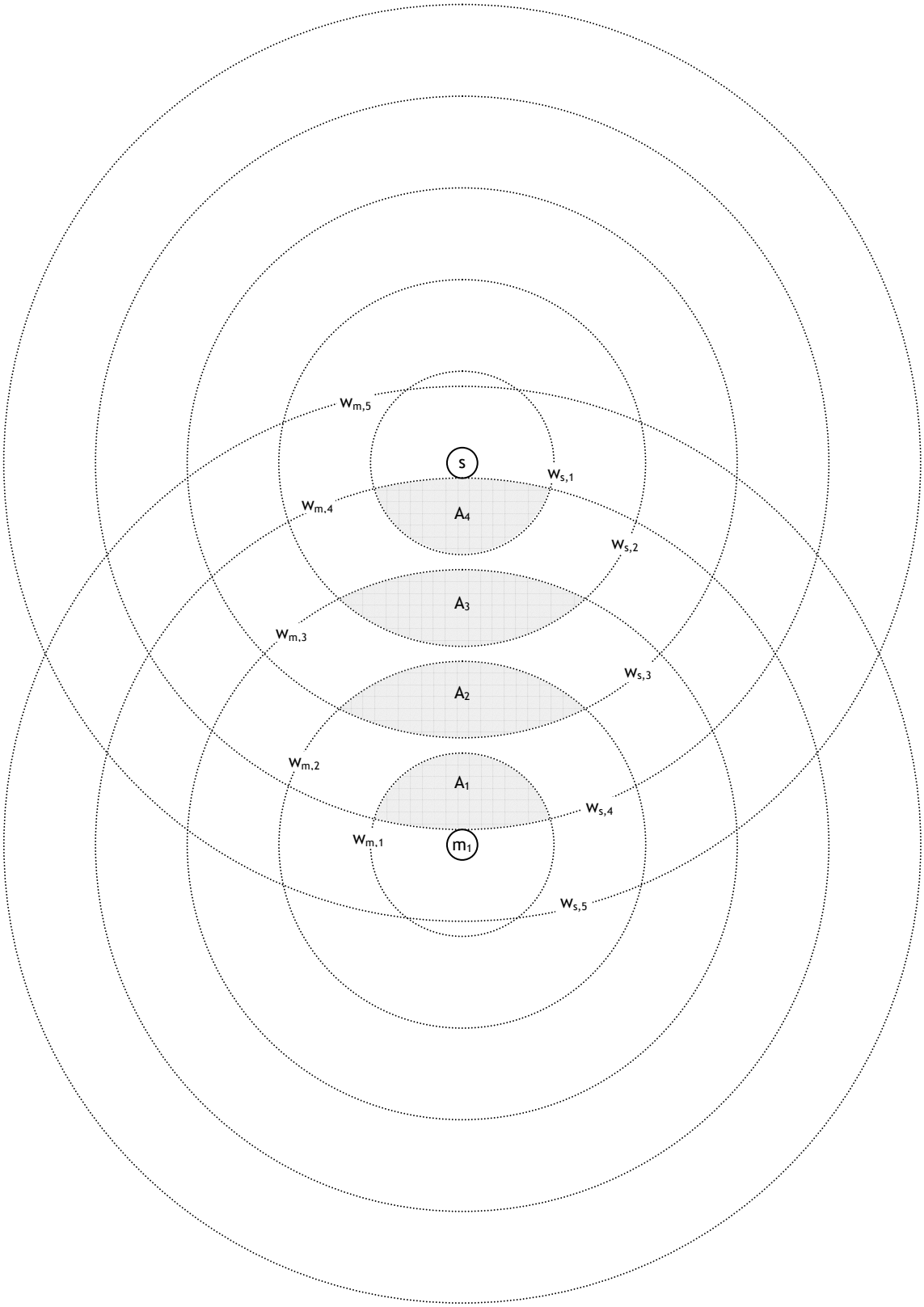



Figure 6.8. The propagation of a join request from the requester towards the server

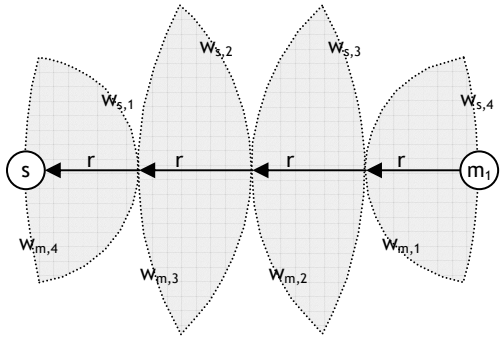



Figure 6.9. The areas containing the nodes involved in the join process

In order to find the total number of nodes involved in the join process, the sum of the four areas covering these four groups of nodes, namely $A_1$, $A_2$, $A_3$ and $A_4$, have to be calculated. The above example is redrawn in Figure 6.10 to show the intersecting areas to be calculated with their $x$ values along the axis.

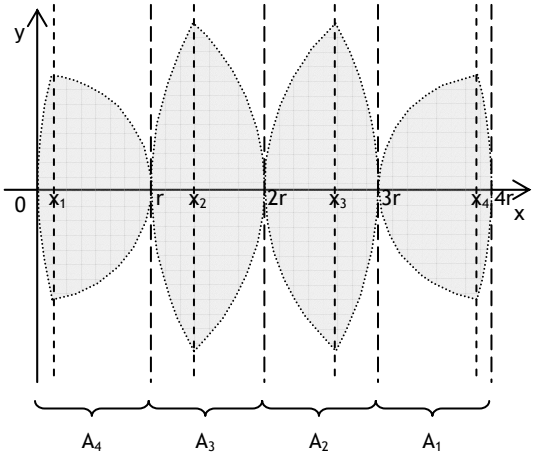



Figure 6.10. Integral boundaries of the areas involved in the join process

In order to generalize the case for the join operation, the total area size has to be calculated as the sum of the areas covering all the affected intermediate nodes at each hop:

$$A_J = \sum_{h=1}^{h_1-1} A_h \tag{6.48}$$

where $h_1$ is the number of hops from the receiver $m_1$ to the server $s$. In other words, the number of intermediate regions between and $m_1$ and $s$ is one less than the number of hops between them.

There are two groups of circles to be formulated for the calculation of the integrals. The first group, which are centred at the server *s*, have the following common equation:

$$x^2 + y^2 = h^2 r^2 \tag{6.49}$$

where $1 \le h \le h_1$-1.

The second group of circles centred at the requester $m_1$ is formulated as follows:

$$\left(x - [h_1 - 1]r\right)^2 + y^2 = \left(h_1 - h\right)^2 r^2 \tag{6.50}$$

where $1 \le h \le h_1$-1.

To formulate the integrals and compute the areas $A_1$, $A_2$, $A_3$, $A_4$, the integration boundaries have to be calculated by eliminating $y$ from Equation 6.49 and Equation 6.50. The equations of each pair of intersecting circles are solved for $x_h$ as follows:

$$h^2 r^2 - x_h^2 = \left(h_1 - h\right)^2 r^2 - \left(x_h - [h_1 - 1]r\right)^2 \tag{6.51}$$

$$x_h = \frac{r}{2} \frac{2h_1(h-1)+1}{h_1 - 1} \tag{6.52}$$

where $1 \le h \le h_1$-1.

Thus, the $x_h$ value of the intersection is a function of the current hop $h$. The total area involved in the join process, $A_J$, which is the sum of all the integrals $A_h$, is:

$$A_J = 2\sum_{h=1}^{h_1-1} \int_{(h-1)r}^{x_h} \sqrt{\left(h_1 - h\right)^2 r^2 - \left(x - [h_1 - 1]r\right)^2}\, dx + \int_{x_h}^{hr} \sqrt{h^2 r^2 - x^2}\, dx \tag{6.53}$$

where $1 \le h \le h_1$-1 and $x_h$ is determined for each term using the function given in 6.52.

Using the node density $\rho$ as given in Equation 6.1, $\nu_J$, which is the number of nodes within the area $A_J$ involved in the join request of $m_1$, can be found:

$$\nu_J = \rho A_J \tag{6.54}$$

It is known that the join process consists of the request, reply and reserve phases in AQM. The request and reply packets are forwarded by $\nu_J$ nodes towards the server as explained above, whereas the reservation packets in the last phase are aimed at exactly one selected upstream node. Thus, the total number of control messages $\mu_J$ processed by the intermediate nodes is:

$$\mu_J = 2\nu_J + h_1 - 1 \tag{6.55}$$

Using this formula with the expected number of hops a receiver needs to join a multicast tree, it is possible to estimate the control overhead of a typical join operation or the control overhead per session, per member, per time unit.

With the help of the symmetry of the shape along the axis crossing the midpoint of the $s - m_1$ line at $\frac{(h_1 - 1)r}{2}$ and applying substitution techniques, $A_J$ can be exactly determined. However, since both the sum of integrals as well as the intervals of each integral also depends on $h$, it is preferable to use one of the approximations presented in Figure 6.11 for the calculation of the area involved in the join process.

It can be argued that for a small number of hops, the approximations with the rectangular areas are more appropriate. However, the ellipse is the only shape that covers all of the partial areas regardless of $h_1$, which makes it the preferred approximation to find an upper bound for the total area. The definition of an ellipse is the set of points in a plane whose distances from two fixed points in the plane have a constant sum. Since the sum of the radii of the intersecting expanding rings centred at $s$ and $m_1$ is always $h_1$, it is obvious that the ellipse always covers the areas $A_1$, $A_2$, $A_3$, $A_4$.

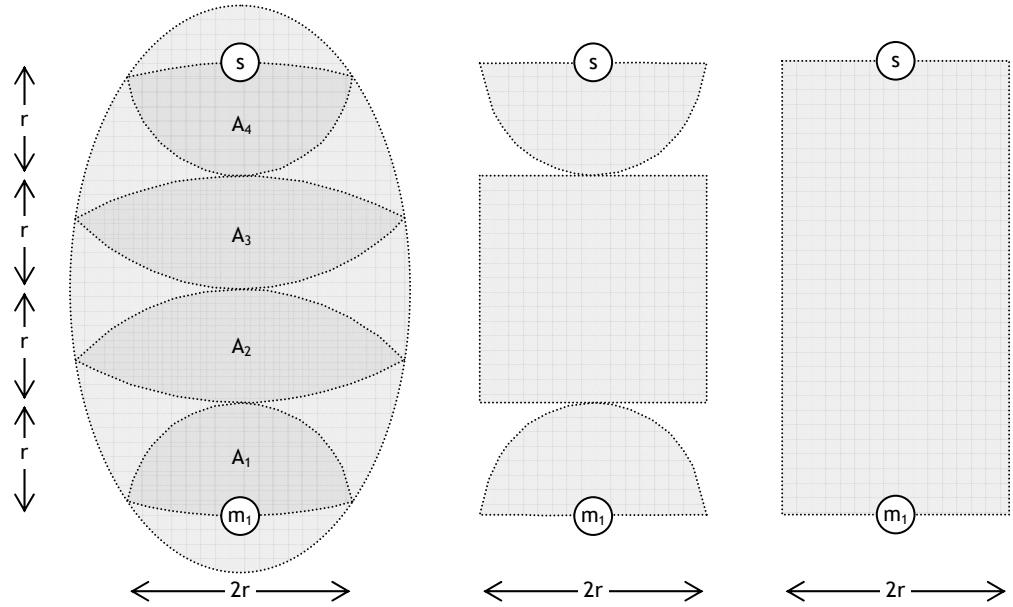



Figure 6.11. Approximations to the area involved in the join process

The area of the ellipse can be formulated as follows:

$$A'_J = \frac{\pi r^2}{4} h_1 \sqrt{2h_1 - 1} \tag{6.56}$$

The area of the rectangle can be formulated as follows:

$$A''_J = 2(h_1 - 1) r^2 \tag{6.57}$$

Selecting the ellipse as the approximation to the area involved in the join request is also useful for smoothing away the decisional errors made by some of the nodes. Since the session update messages have a certain period, it is possible that there are nodes that react to join requests although they should not. These nodes may have lost their connection to the session or moved away from the location where they have been able to support it. In other words, the topological changes in the network, which is a result of node mobility as well as the properties of the wireless medium, may cause some of the nodes that are actually outside the involved area to take part falsely in the process. Thus, an ellipse reaching from the requester to the session server is a logical approximation to represent the propagation of the control messages between these two nodes.

## 6.4. Interpretation of Results

An analysis which thoroughly examines the join process of AQM is presented. First, with the help of a few simplifying assumptions, the probabilities of reaching the server in a certain number of hops are calculated for the first receiver of a session, which is given in Equation 6.8. Then the same probabilities are analysed for the second join attempt, the result of which is given in Equation 6.45. These computations are based on the ratio between a node's transmission range and the network area. Finally, the join process of a single AQM node is examined in isolation. The propagation of the join request is analysed in order to find an average value for the number of intermediate nodes involved. Using this result, which is formulated in Equation 6.54, it is possible to find the number of packets propagating in the network, which is given in Equation 6.55, during a typical join process. Combining these equations with Equation 6.8 and Equation 6.45, more general results can be obtained such as the average or worst-case control overhead per session, per member, or per unit time throughout the network.

The results presented in Section 6.1 and Section 6.2 show that, after the first member of a session connects to the server, subsequent join requests can be fulfilled in fewer hops, requiring a lower control overhead. Figure 6.12 and Figure 6.13 depict the individual and cumulative probabilities for each hop using Equation 6.8 and Equation 6.45 for the first two session receivers in a network where $\gamma = 10$. The probabilities of the second receiver are calculated by using the average hop count of the first receiver as the value of $h_1$ in Equation 6.45. The probabilities for lower hop counts are higher for the second receiver than the first, which means that it has a lower average hop count. It can be argued that, since the coverage area increases with each new member, this trend continues for subsequent receivers.

As depicted in Figure 6.5 and Figure 6.6, two extreme cases are shown for the area covered by the multicast tree after the first session member, where the average distance between two consecutive nodes is $r$ and $\frac{r}{2}$, respectively. Equation 6.45 and Equation 6.46 use these values to find the probabilities for the number of hops required by the second member to join the session in these two cases.

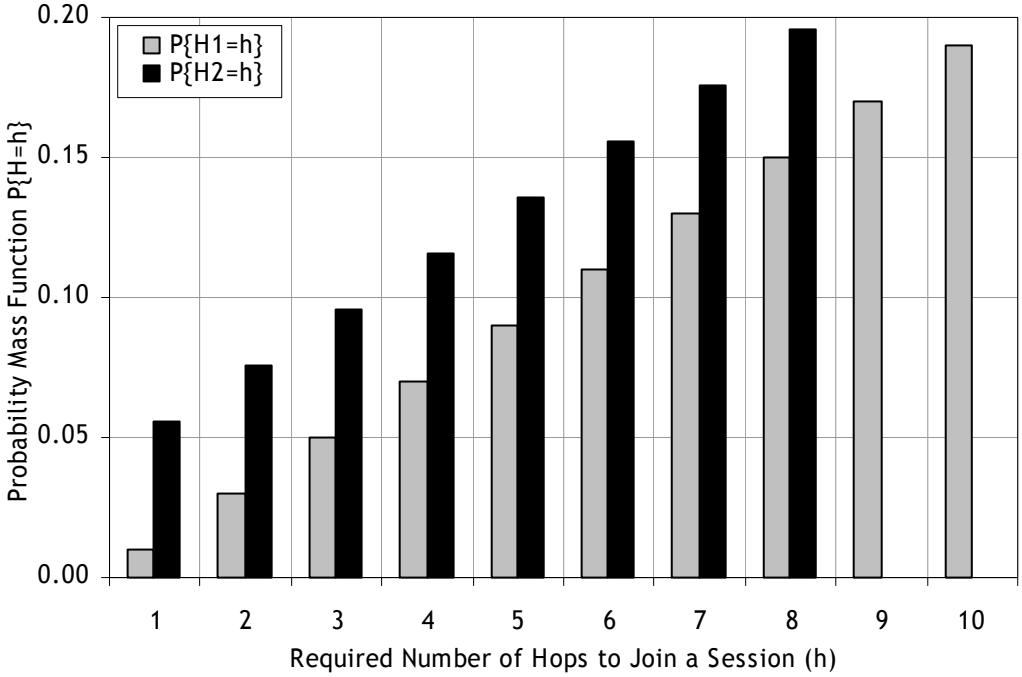



Figure 6.12. Hop count probabilities for the first and second receivers

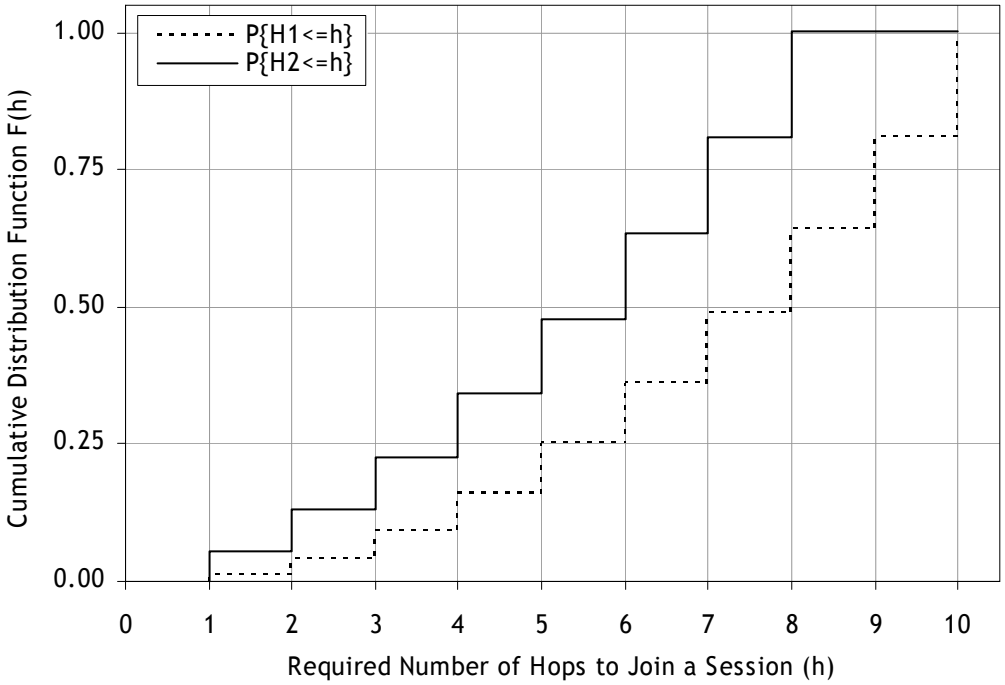



Figure 6.13. Probability distribution of the hop count for the first and second receivers

Figure 6.14 shows how the number of hops required by the first member to join the session affects the join process of the second member in a network where $\gamma = 10$, which is the ratio between the network radius *R* and the transmission range *r*. As mentioned in the previous paragraph, the expected values for the second receiver are computed from their respective probabilities in two extreme cases. First, the value of $P\{H_1 = h\}$ is calculated using Equation 6.8 for each possible $h_1$ where $1 \le h_1 \le 10$. Then, $P'\{H_2 = h\}$ and $P''\{H_2 = h\}$ are found by using the $P\{H_1 = h\}$ values in Equation 6.45 and Equation 6.46, respectively. Finally, the expected values are determined. The two cases are shown in the legend with their average node distances *r* and $\frac{r}{2}$, respectively. It can be seen that the average hop count required by the second member decreases as the path between the first member and the server becomes longer. This result confirms that the increase in the number of intermediate nodes leads to a larger aggregate coverage area, which makes it easier for subsequent members to reach the tree. The average hop count of the first receiver, which is formulated by Equation 6.13, is drawn for comparison purposes. It can be argued that the trend of decreasing average values continues for the subsequent members of the session. Therefore, the results achieved for the second receiver can be used as worst-case values for the rest of the receivers.

Figure 6.15 shows how the average number of hops for members to join a session increases as $\gamma$ increases. The calculations are similar to those explained above. The difference is that the expected values of the hop count of the first receiver is used to determine the averages for the second receiver for different values of $\gamma$, where $1 \le \gamma \le 10$. As expected, a node needs more hops to reach the server on the average as $\gamma$ grows. However, there is a higher rate of change in the average number of hops for the first nodes to join the tree, which confirms once again that the second and later receivers require fewer hops for the same session. By showing that the expected hop distance to the multicast tree decreases with each additional receiver joining the session, it can be concluded that the multicast tree gets probabilistically closer for the next member.

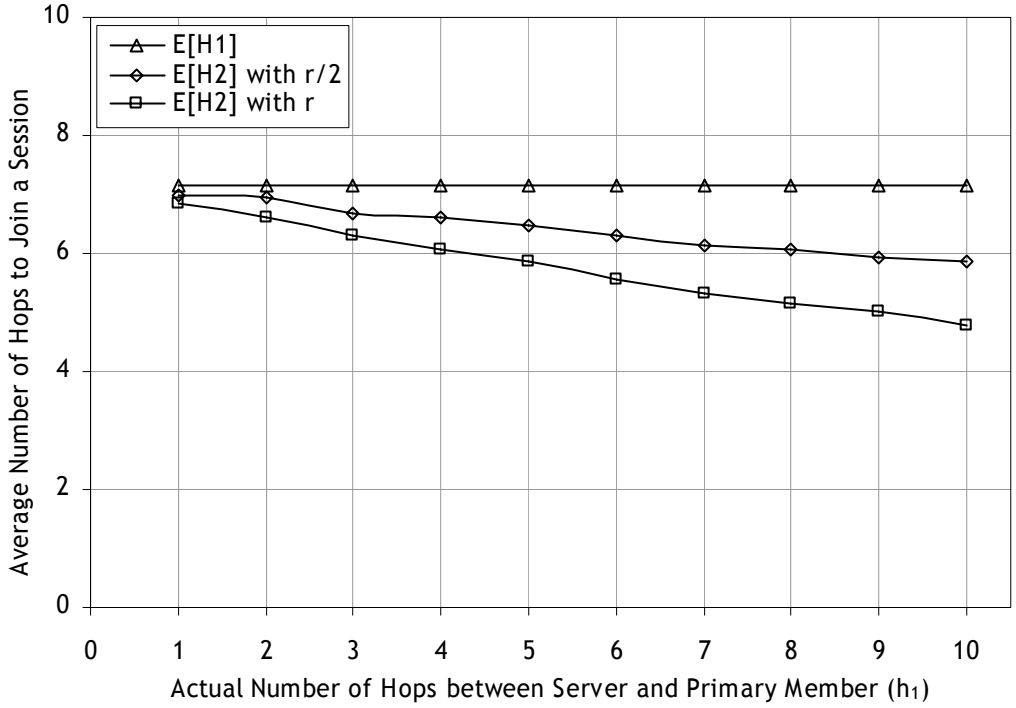



Figure 6.14. Expected value of the hop count for the first and second receivers

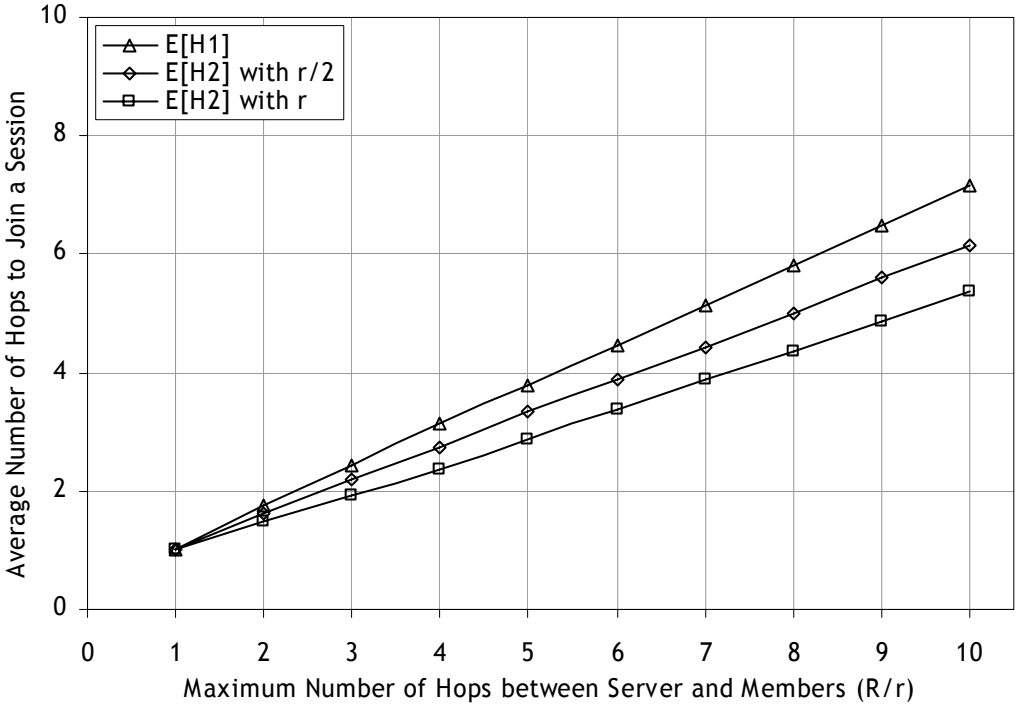



Figure 6.15. Expected value of the hop count as a function of increasing $\gamma$

The analytical results presented in Figure 6.15 can be used to derive the number of nodes involved in a join process with the help of Equation 6.54. Equation 6.56 and Equation 6.57 formulate the ratio of the propagation area of a join request to the area of the network as a function of the number of hops between the requester and the server. Once this is done, it is easy to derive the number of processed control packets via Equation 6.55.

A new set of simulations are conducted in order to validate the analytical results using OPNET Modeler 10.5 Educational Version and Wireless Module [100]. Simulations are repeated 40 times for each data point and results are aggregated with a 95 per cent confidence interval. Nodes are placed randomly in a circular area. Only one server is allowed, which is placed at the centre of the circle. Mobility is omitted. Other simulation settings are presented in Table 6.1. In order to achieve constant node density, the network population is kept proportional to $R^2$.

Table 6.1. Simulation settings for the analysis of the join process

| $\frac{R}{r}$ | Number of Receivers | Network Radius | Transmission Range |
|---|---|---|---|
| 2 | 25 | 500 m | 250 m |
| 3 | 36 | 600 m | 200 m |
| 4 | 49 | 700 m | 175 m |
| 5 | 64 | 800 m | 160 m |
| 6 | 81 | 900 m | 150 m |
| 7 | 100 | 1 000 m | 145 m |

Figure 6.16 compares the number of nodes involved in the join process of the first session member experienced via simulation with the values computed analytically. As mentioned above, the analysis is based on the expected values of the hop distance between the candidate and the server and provides an approximation to the number of intermediate nodes in the propagation area of the join request as given in Equation 6.57. The simulation results are below these values but follow the trend suggested by the analysis.

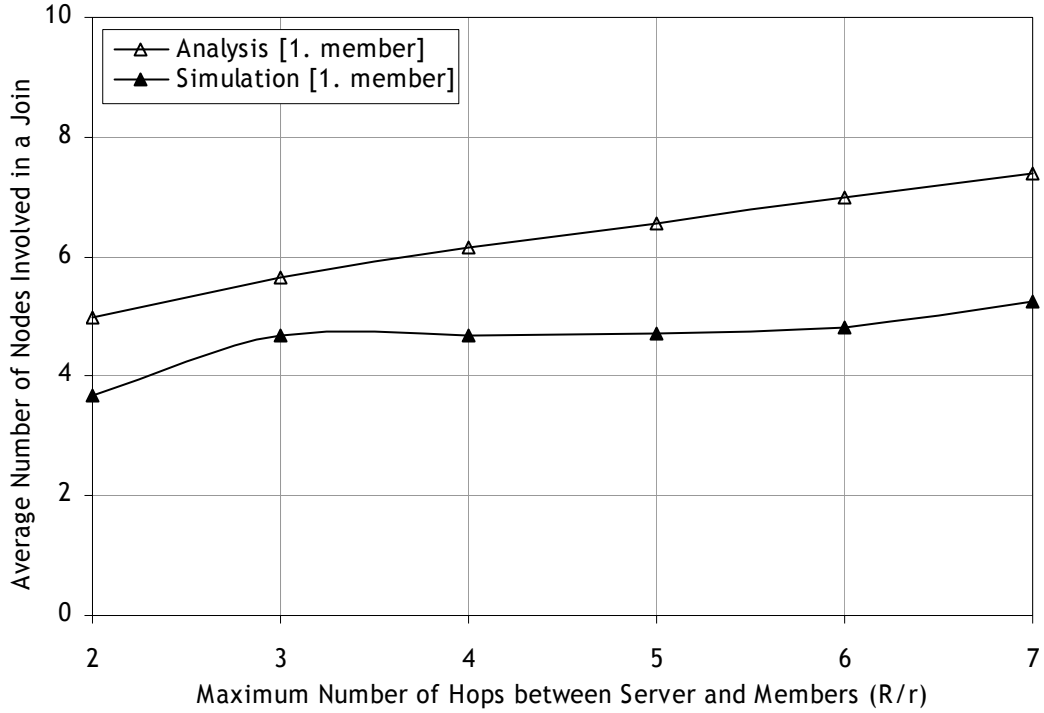



Figure 6.16. The number of nodes involved in the join process of the first receiver

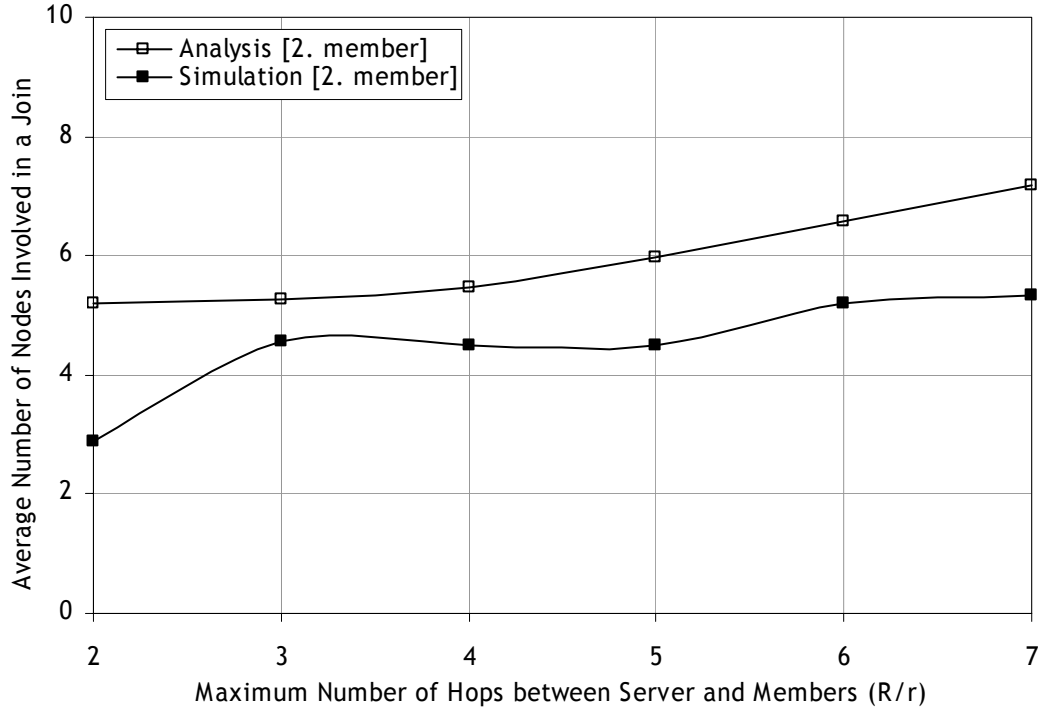



Figure 6.17. The number of nodes involved in the join process of the second receiver

Figure 6.17 makes the same comparison for the second join attempt after the initiation of the multicast tree between the server and the first member. This time, however, Equation 6.56 is used to approximate the size of the propagation area with slightly looser bounds. This way, the possible divergence at the boundaries of the area due to the mobility of multiple nodes building the initial multicast tree is covered better. It can be seen that the expected values of the analysis as well as the results achieved by simulation are below the averages of the first member. Thus, it is shown both analytically and experimentally that the overhead of a join attempt by subsequent candidates decreases as more nodes become session members. On the other hand, the simulation results are closer to the analytical results and follow the same trend which suggests that the number of nodes taking part in the join process increases gracefully as the size of the network grows.

Although the analysis presented in this section does not provide tight bounds for the control overhead incurred by AQM, it gives sufficient insight into the behaviour of the protocol. The length of the propagation area of a typical join request depends on the number of hops between the server and the new member, which is calculated as an expected value and used to determine the average number of nodes involved in the join process. According to the analysis results, which are also validated through simulation, AQM provides a session and membership management system to its users, whereby each new session member can join the multicast tree with an acceptable overhead, which degrades gracefully throughout the session.

As mentioned previously, the registration of extra forwarders by the receivers during multicast data streaming is not taken into consideration since the tree-to-mesh evolution presented in Section 3.4.2 does not affect the analysis. It can be seen easily that the analysis is actually still valid with the mesh option since the extra forwarders to be registered are already members of their respective sessions and the join process of a subsequent receiver remains the same. In other words, the tree-to-mesh evolution does not affect the results of the analysis due to the fact that only new wireless links, but no new nodes are added to the multicast tree during this operation, which means that the coverage area of the multicast group does not change.

# 7. AN IMPROVEMENT IN ADMISSION CONTROL: OBJECTION QUERIES

The performance evaluation presented in Section 5 shows that AQM is superior to the non-QoS protocol regarding the satisfaction of session members in general. However, it should be examined if and how AQM can achieve a still higher QoS sustainability ratio. By definition, QoS sustainability decreases as a result of dropping those members which experience excessive delay or data loss. Both of these symptoms can be kept at lower rates if more accurate reservation decisions can be made by the upstream members forwarding data to the nodes being dropped. In other words, the network can maintain a higher QoS level if intermediate session members can protect themselves from too many allocations. Otherwise, even a small number of overloaded members can affect many concurrent flows, which can yield overloaded sessions and cause performance degradation when the network density or the traffic load is high. To overcome this problem, an extension to AQM is designed, whereby each node about to send a reply during the join process of another node consults its neighbourhood to see if there is any potential violation of resource limits.

As mentioned previously, a node decides whether or not to take part in a session as a forwarder based on its current resource availability. The approach to the calculation of residual bandwidth, which is presented in Section 4, provides each node with a sufficient method to estimate the bandwidth availability within its neighbourhood individually. However, it does not consider the bandwidth allocations beyond direct neighbours. Thus, the approach helps a node protect itself, but is not enough to prevent that node's neighbours from becoming overloaded. Although a node does not allocate more bandwidth than available in its neighbourhood, it can experience an overload problem as a result of the allocations made by its neighbours which cannot directly detect each other in the wireless medium, where resources are shared.

In summary, an AQM node has to ensure that, by allocating bandwidth to a new request, it does not cause one or more of its neighbours to suffer from overload as a result of excessive bandwidth usage in their neighbourhood. The objection query mechanism is developed to serve this purpose, which is introduced in the following sections.

### 7.1. Overloaded Neighbourhood Problem

A node can be surrounded by several neighbours, some of which are not within the transmission range of each other. Looking from the viewpoint of the node in the centre of that neighbourhood, it is possible that two of its neighbours not aware of each other can allocate resources in such a way that neither of them violates the QoS requirements within its own neighbourhood, whereas their total allocation exceeds the capacity limit of their common neighbour. In this case, the node in the centre experiences overload due to excessive resource allocation in its neighbourhood, which can be neither foreseen nor prevented since the surrounding nodes are not informed about each other's reservations. Thus, a particular kind of hidden terminal problem prevents nodes from making more accurate reservation decisions.

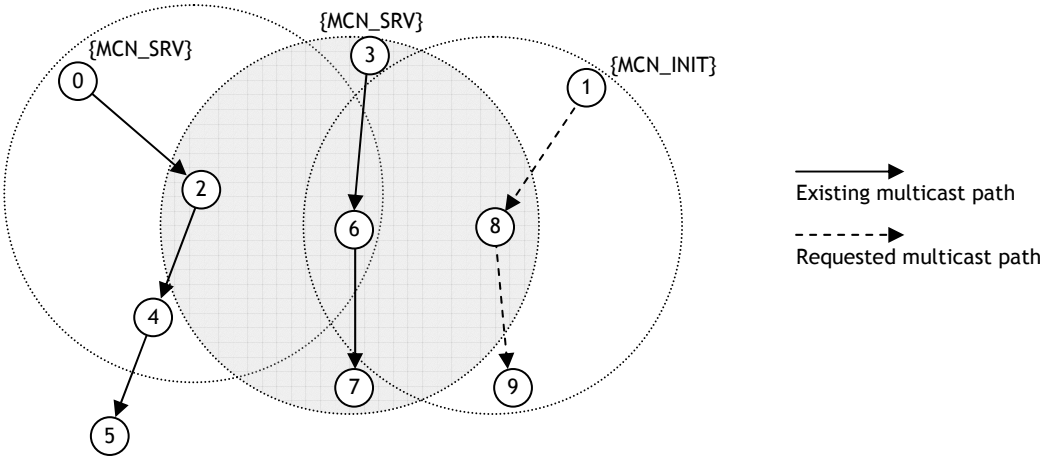



Figure 7.1. The overload problem in the neighbourhood of $n_6$

Figure 7.1 illustrates the overloaded neighbourhood problem caused by nodes that are unaware of each other's resource utilization. It is assumed that there are two ongoing sessions, which do not cause any overload in the network. The first session is served by $n_0$ with $n_2$ as a forwarder, $n_4$ as a forwarding receiver and $n_5$ as a receiver, whereas the second session is served by $n_3$ with $n_6$ as a forwarder and $n_7$ as a receiver. Then $n_9$ sends a join requests for the session of $n_1$, which is forwarded by $n_8$ and eventually replied by $n_1$. The reply also propagates over $n_8$ and reaches $n_9$, which selects this path to actually join the session and sends the reservation message upstream. Looking from the viewpoint of $n_8$, the reservation does not present any problems. Since the recent greeting messages it has

received from its neighbours show that the total resource allocation in its neighbourhood allows the acceptance of the new session, it allocates the required bandwidth and starts forwarding the multicast data. From the viewpoint of $n_6$, however, there are already three data forwarding nodes in its neighbourhood, namely $n_2$, $n_3$ and $n_6$ itself, using the available bandwidth of the neighbourhood to its limits. Thus, the extra bandwidth allocation of $n_8$ causes $n_6$ to experience overload in its neighbourhood, mainly due to the fact that $n_8$ is not aware of the bandwidth consumption of nodes $n_2$ and $n_3$.

### 7.2. Objection Query Mechanism

To overcome the overloaded centre node problem, an extension to the request-reply-reserve process is necessary, whereby each replying node first consults its neighbours to see if any of them becomes overloaded. In other words, a node has to forward a reply only after querying its neighbourhood and making sure that none of its neighbours will become overloaded as a result of a resource allocation made for the session in question, in case the node will be on the path chosen by the requester.

According to the objection query mechanism, a node about to forward a reply first sends an objection query to its immediate neighbours. This is a one-hop message containing information on the requested bandwidth, which warns all neighbours to check whether they become overloaded as a result of this potential allocation. If the new reservation causes its limit to be exceeded, the neighbour sends an objection to the node which has queried it. Otherwise the query is discarded. If the node having originated the query receives any objection, it cancels the forwarding of the reply. Otherwise the query times out, indicating that the reply can be sent safely. Only those neighbours who are serving one or more sessions may object to new allocations. It is not important that an inactive node becomes overloaded. Therefore, such a node discards objection queries even if the requested bandwidth causes overload in its neighbourhood. The objection query mechanism is used by each node preceding the downstream flow of the reply packet in time. The first replier of a request can be an active member of the session forwarding data packets, such as a server, a forwarder or a forwarding receiver. These nodes do not need to send objection queries since they must have been through the same process previously, consulted their neighbours and allocated the necessary resources successfully. Although a

receiver is also counted as an active session member, it has to send an objection query to its neighbours since, unlike the others, it has not allocated any resources previously. A session initiator, which is about to reply the request of its first member, or an intermediate node, which is about to forward a reply towards the requester as a predecessor also has to issue an objection query.

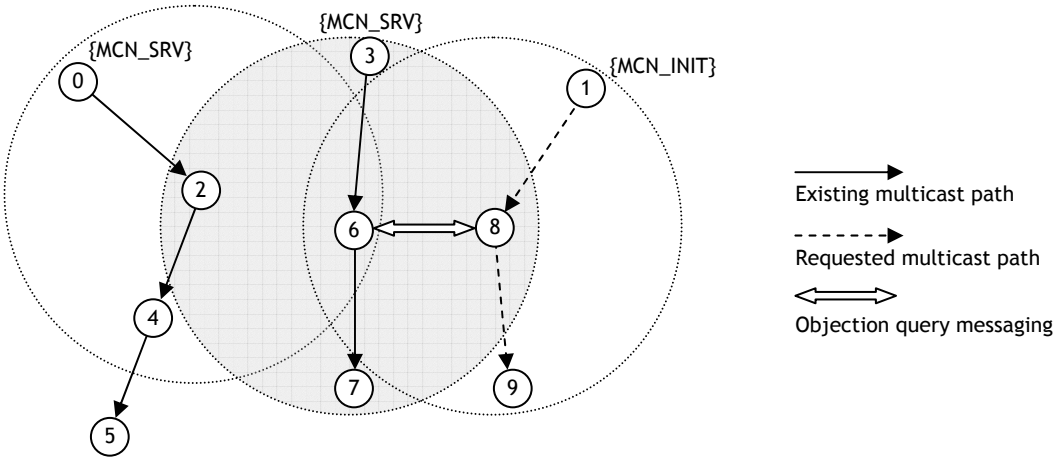



Figure 7.2. The objection query received and responded by $n_6$

Figure 7.2 illustrates the utilization of the objection query mechanism, whereby the reply phase of the join process described in Figure 7.1 is revisited. In this case, the nodes $n_1$ and $n_8$ send one-hop objection queries before sending their replies. The figure focuses on the step where $n_8$ receives the original reply from $n_1$ and is about to decide whether or not to forward it further downstream. Thus, $n_8$ has to look for objections from within its neighbourhood. Therefore it issues an objection query. The one-hop message is received by its neighbours $n_1$, $n_9$ and $n_6$. The former two ignore the query since they have already made the necessary controls as members of the same multicast session. The primary concern of $n_8$ by sending the query is to inform those nodes in its neighbourhood that are not part of the path being built. The objection query allows $n_6$ to object to a possible allocation made by $n_8$, if it starts suffering from overload, which is already sharing its neighbourhood bandwidth with $n_2$ and $n_3$. Thus, $n_6$ sends an objection to $n_8$ if overload occurs or discards the query if there is enough free bandwidth to allow the new data flow.

There are several alternatives with regard to the implementation of the objection query mechanism. In a possible implementation, the objections can be postponed to the

reserve phase. Nodes which overhear the reservation message aimed at another node can object to this allocation if the node to reserve resources is also within their neighbourhood and causes them to be overloaded. In this case, it is necessary to inform the requester with an error message that the final reservation has failed and it should select another path from the reply messages it has received. In contrast to the original objection query mechanism, this is a reactive approach. Whether the proactive approach of the reply phase or the reactive approach of the reserve phase performs better in terms of control overhead depends on the general traffic characteristics of the ad hoc network. The proactive approach is beneficiary under heavy traffic load. Otherwise, the reactive approach is more efficient. In the next section, the details are given for the alternative where the mechanism is implemented proactively in the reply phase as explained above.

### 7.3. Modifications on the Membership Module

The implementation of the objection query mechanism necessitates an additional field in the table of requests (TBL_REQUEST) to hold the objection status, such as query sent, objection received and objection timeout, as well as an additional control packet for sending and receiving the objection queries (JOIN_OBJ). As mentioned before, the mechanism is utilized in the reply phase of the join process. Thus, two procedures need revisions, which handle the reception of join requests and replies.

Similar to the procedure defined in the basic AQM protocol, a forwarded request eventually reaches some nodes which are already members of that session and therefore can directly send replies (JOIN_REP) back to the requester. Downstream nodes that have forwarded join requests start receiving the corresponding replies from these members. Since it is possible to qualify as forwarders later, they send objection queries (JOIN_OBJ) to their immediate neighbours before replying. Such a query is necessary to check whether a possible resource allocation violates the bandwidth limitations of the neighbours. It is otherwise impossible for a node to see the bandwidth usage beyond its neighbours and to know if they can become overloaded. If no objections are returned by the neighbours in a predefined amount of time, the nodes combine the QoS information with the QoS they can currently offer and send the updated JOIN_REP towards the requester. The information on the originator and the immediate forwarder of the reply are kept in the JOIN_REP packet.

```
Read JOIN_REQ packet received from network module;
IF ((known session) && (process status == NULL) &&
     (own hop count < packet sender hop count))
     {
     Calculate neighbourhood bandwidth allocation;
     Calculate available bandwidth;
     IF ((available bandwidth >= session bandwidth) &&
        (own hop count <= session hop limit) &&
        (own node state == MCN_PRED))
          {
          Enter JOIN_REQ data into TBL_REQUEST;
          Update process status [request forwarded] in TBL_REQUEST;
          Forward JOIN_REQ packet;
          Set expiration timer for JOIN_REQ;
          }
     ELSE IF (((own node state == MCN_INIT) ||
        (own node state == MCN_RCV)) &&
        (available bandwidth >= session bandwidth) &&
        (own hop count <= session hop limit))
          {
          Enter JOIN_REQ data into TBL_REQUEST;
          Update process status [request received] in TBL_REQUEST;
          Send JOIN_OBJ packet;
          Update objection status [query sent] in TBL_ REQUEST;
          Set objection timer for JOIN_OBJ;
          }
     ELSE IF ((own node state == MCN_SRV) ||
        (own node state == MCN_FWD) ||
        (own node state == MCN_FRCV))
          {
          Enter JOIN_REQ data into TBL_REQUEST;
          Send JOIN_REP packet;
          Update process status [reply forwarded] in TBL_REQUEST;
          }
     }
```

Figure 7.3. Revised procedure for the reception of a join request

Figure 7.3 displays the revisions made to the pseudocode for the handling of the JOIN_REQ packets by the intermediate nodes. Considering the case where the changes to the procedure take place, a MCN_INIT waiting for its first receiver or a MCN_RCV which receives multicast data but does not forward them have to check for necessary QoS conditions since they have not allocated any resources yet. Even if QoS requirements can be met, however, they still have to ensure that their neighbours are not overloaded. Thus, they send one-hop JOIN_OBJ packets to their neighbours to see if there are any objections. They also set a timer to limit the time they will wait for objections. Finally, a MCN_SRV, a MCN_FWD or a MCN_FRCV already sending multicast data to their receivers, can respond to the request directly with a JOIN_REP as usual. In all these cases, a record is entered to TBL_REQUEST to show the current status of the request.

```
Read JOIN_REP packet received from network module
IF ((process status == request forwarded) && (NOT expired JOIN_REQ))
     {
     Reset expiration timer for JOIN_REQ;
     IF (packet sender state == NULL)
         (packet sender's hop count < own hop count))
          {
          Enter packet sender [state = MCN_PRED] into TBL_MEMBER;
          IF (packet sender's hop count + 1 < own hop count)
               {
               Update own node [packet sender's hop count + 1]
                                                    in TBL_MEMBER;
               }
          }
     IF (own request)
          {
          Update packet sender [role = upstream node] in TBL_REQUEST;
          Update packet sender [state = MCN_FWD] in TBL_MEMBER;
          Update own node [state = MCN_RCV] in TBL_MEMBER;
          Update process status [reserve forwarded] in TBL_REQUEST;
          Send JOIN_RES packet to packet sender;
          Report successful join to application module;
          }
     ELSE
          {
          Calculate neighbourhood bandwidth allocation;
          Calculate available bandwidth;
          IF ((available bandwidth >= session bandwidth) &&
             (own hop count <= session hop limit))
            {
            Update packet sender [role = upstream node]
                                                  in TBL_REQUEST;
            Update process status [reply received]
                                                  in TBL_REQUEST;
            Send JOIN_OBJ packet;
            Update objection status [query sent] in TBL_REQUEST;
            Set objection timer for JOIN_OBJ;
            }
          }
     }
```

Figure 7.4. Revised procedure for the reception of a join reply

Figure 7.4 shows the revisions made to the pseudocode for the handling of the JOIN_REP packets. Focusing only on the changes, if an intermediate node receives a JOIN_REP which corresponds to a JOIN_REQ it has forwarded earlier, it has to send its own JOIN_OBJ to check its neighbourhood before forwarding the received JOIN_REP. If the necessary QoS conditions exist, the node updates the record in its TBL_REQUEST and sends the one-hop JOIN_OBJ packet to its neighbours to prevent any possible overload from occurring. It also sets a timer, at the end of which it finally forwards the JOIN_REP packet if it has not collected any objections in response to its query.

```
Read JOIN_OBJ packet received from network module
IF (own bandwidth allocation)
     {
     Calculate neighbourhood bandwidth allocation;
     Calculate available bandwidth;
     IF (available bandwidth < objection query bandwidth)
          {
          Send back JOIN_OBJ packet;
          }
     }
```

Figure 7.5. Procedure for the reception of an objection query

Figure 7.5 displays the pseudocode for the handling of the JOIN_OBJ queries. Upon reception of the packet, each node checks if it has any previous bandwidth allocation, which is the simplest way to determine whether the node has an active role in any session. If this is the case, the node compares the available bandwidth to the possible allocation related with the query and objects to this allocation if it conflicts with the resource limits.

```
Read JOIN_OBJ packet received from network module
Update process status [objection received] in TBL_REQUEST;
Update objection status [query objected] in TBL_REQUEST;
```

Figure 7.6. Procedure for the reception of an objection in response to a query

Figure 7.6 and Figure 7.7 show how the originator of the objection query handles the process. If it receives an objection within the defined timeout, it updates its records. At the end of the timeout, which is the deadline of waiting for objections, it checks these records to see whether it has received any JOIN_OBJ packets in response to its query. If this is the case, the JOIN_REP packet is not sent downstream any further since the node cannot allocate resources for this session without causing overload in its own neighbourhood.

```
IF (objection status != query objected)
     {
     Send JOIN_REP packet;
     Update process status [reply forwarded] in TBL_REQUEST;
     }
ELSE
     {
     Delete request record from TBL_REQUEST;
     }
```

Figure 7.7. Procedure for the completion of the query with the objection timeout

## 7.4. Performance Evaluation

The evaluation of AQM with the addition of objection queries presented in this section is based partly on a selection of the performance metrics introduced in Section 5.1, which are related to member satisfaction and network load. The simulation settings are identical to what is summarized in Section 5.1.2, with the QoS classes and simulation parameters as given by Table 5.1 through Table 5.3.

As mentioned in the introduction of this section, one of the purposes of this evaluation is to examine the relation between $Q_{Member}$, $A_{Member}$ and the resource allocation strategies of AQM during a join process in more detail. Thus, the terms overloaded member and overloaded session are introduced, which require clearer definitions such that they can be used in the performance evaluation.

An important aspect of the QoS-related multicast routing decisions made by AQM is the improvement in the ratio of overloaded member nodes, which has a direct impact on the satisfaction of session members regarding the multicast service provided. It is one of AQM's main concerns that network resources are not excessively utilized to avoid excessive delay and packet loss due to overload and keep the QoS conditions at a satisfactory level. Thus, the member overload ratio $O_{Member}$ is formulated as follows:

$$O_{Member} = \frac{o}{z + f} \tag{7.1}$$

where $o$ represents the number of overloaded nodes, which have allocated their resources to serve and forward more sessions than is possible without exceeding the maximum available bandwidth, $z$ is the total number of session servers and $f$ is the total number of session forwarders. The division gives the ratio of overloaded nodes to all serving and forwarding nodes and represents a member-level rate of overload. In addition to being a criterion for member satisfaction, the member overload ratio also gives a good notion of AQM's efficiency in terms of loss and delay, since a node exceeding its bandwidth limit causes these two impacts on itself as well as on its neighbours.

Excessive bandwidth allocation by individual nodes for the course of a session has also an effect on other members of that session. Therefore, it is necessary to observe the implications of these events on sessions as well. The session overload ratio $O_{Session}$ is defined to evaluate the session-level success of AQM to prevent overload and formulated as follows:

$$O_{Session} = \frac{y}{z} \tag{7.2}$$

where $y$ is the number of sessions with at least one overloaded member and $z$, which denotes the total number of servers, is used as the total number of sessions. The term gives the percentage of sessions with one or more overloaded members, which can be interpreted as a session-level overload rate experienced by the ad hoc network. $O_{Session}$ is a more important criterion than $O_{Member}$ since it has a more widespread effect on the network.

The simulations are conducted using OPNET Modeler 11.5 Educational Version and Wireless Module [100]. The mobility model used in the simulations is random waypoint and the simulation environment provided for the improved version of AQM is identical to the one created for the simple version of the protocol. This allows a fair comparison of the results. Simulations are repeated 10 times for each data point and results are aggregated with a 95 per cent confidence interval. The graphics of the improved version are identified in the figure legends with the term AQM-OQ, which stands for AQM with objection queries. The results of the non-QoS protocol are left out in order to avoid repetition and focus on the improvements of AQM. Only the first of the simulation sets presented in Table 5.3, which examines the effect of network density on AQM, is used for evaluation since the degradation in $Q_{Member}$ is most significant when network density grows.

### 7.4.1. Satisfaction of Session Members

Figure 7.8 through Figure 7.13 show the results of the improvement effort regarding the satisfaction of session members in comparison with the original AQM protocol. The general trend observed in AQM-OQ is similar to that of AQM and the relative superiority to the non-QoS protocol is more or less the same.

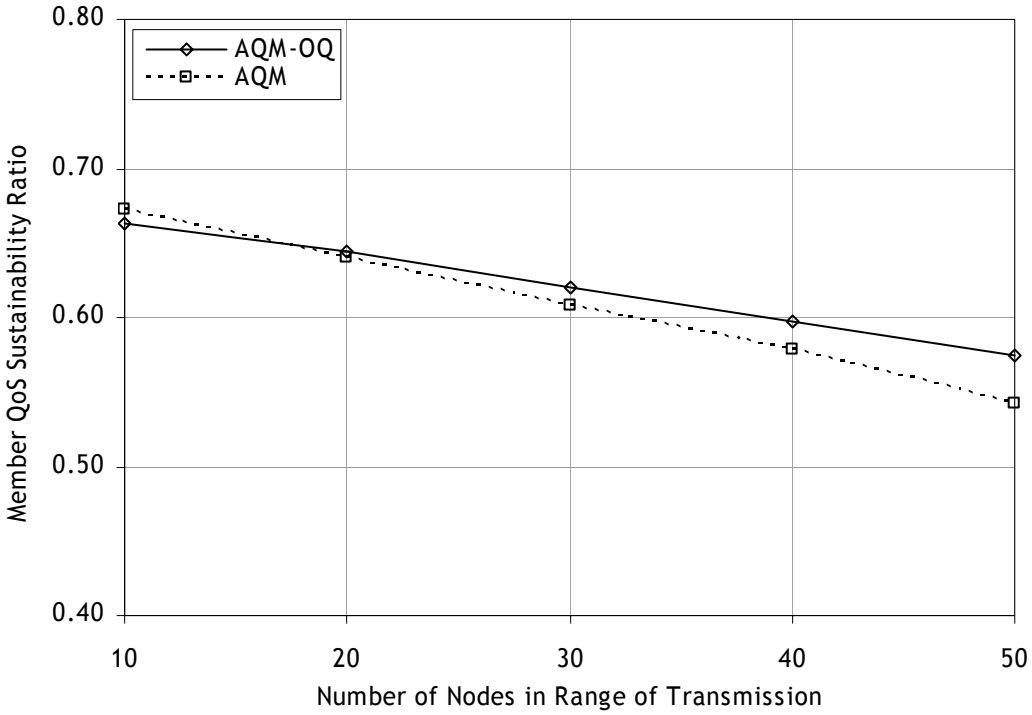



Figure 7.8. Member QoS sustainability as a function of network density

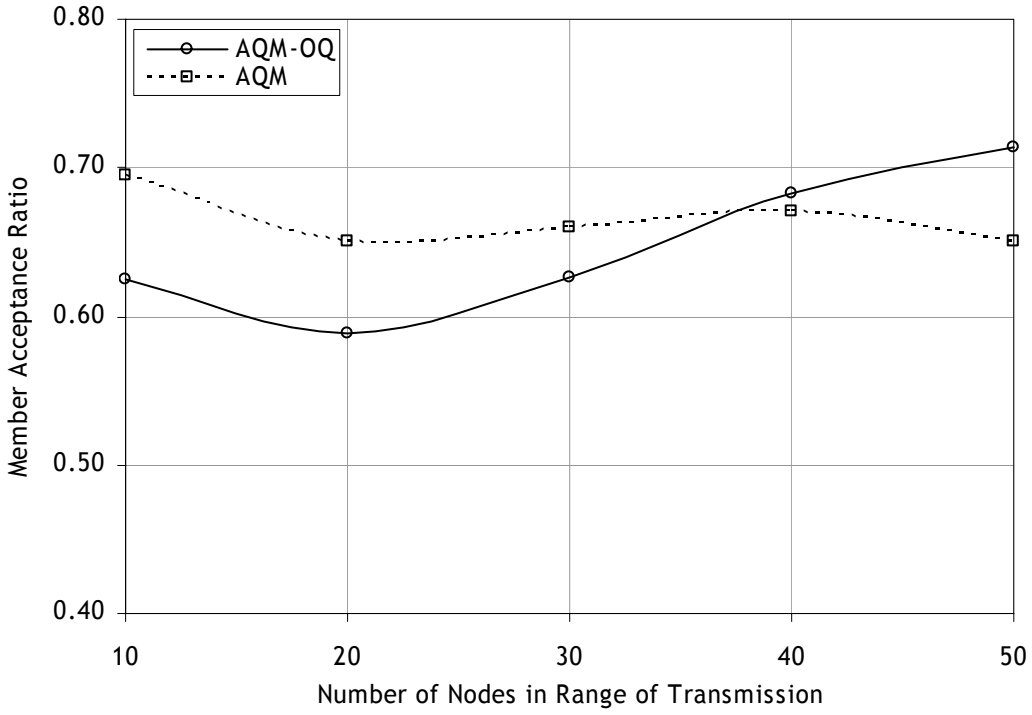



Figure 7.9. Member acceptance ratio as a function of network density

Figure 7.8 presents the difference between the member QoS sustainability ratios of the original and improved versions of AQM with regard to changing network density. AQM-OQ has actually a lower QoS sustainability ratio than AQM when the network is sparse. This is a result of low connectivity, which causes both versions of AQM to drop members in close numbers while AQM-OQ accepts fewer join request due to its more aggressive admission control. However, the objection query mechanism provides AQM-OQ with a slightly higher QoS sustainability than AQM and the difference increases as the network density grows, although the overall QoS sustainability decreases as expected.

Figure 7.9 shows that the member acceptance ratio of AQM decreases when the objection query mechanism is enabled. This is an expected result since it is already known that there is a tradeoff between the sustainability of QoS for session members and the acceptance rate of the join requests. AQM-OQ utilizes a stricter admission control scheme with objection queries to detect potentially overloaded nodes beyond direct neighbours. If the node sending the objection query receives any objection, it cancels the reply. Therefore, it accepts fewer join requests. The acceptance rates of both AQM as well as AQM-OQ are initially high for sparse networks, due to the low connectivity of the nodes. As mentioned previously, AQM nodes join only those sessions that they are aware of. In a sparse network, session initiation and update messages cannot travel very easily. Thus, many nodes are prevented from sending requests since they are not aware of any available sessions. The result is a relatively higher member acceptance ratio, although QoS sustainability is not particularly high due to low connectivity.

As the average number of nodes within a node's transmission range increases, network connectivity reaches a point where more requests are made since a higher portion of session initiation and update messages can reach potential receivers. However, in the attempt to protect session members from overload, objection queries prevent many of these requests from being accepted. Thus, AQM-OQ accepts a lower portion of requests than AQM due to resource constraints. Nevertheless, the acceptance ratio of AQM-OQ improves as the network density grows such that it is actually better than AQM for denser networks, even though QoS sustainability is also improved. This behaviour seems to be unexpected, but can be explained after a more detailed examination of acceptance ratios and QoS sustainability of individual QoS classes.

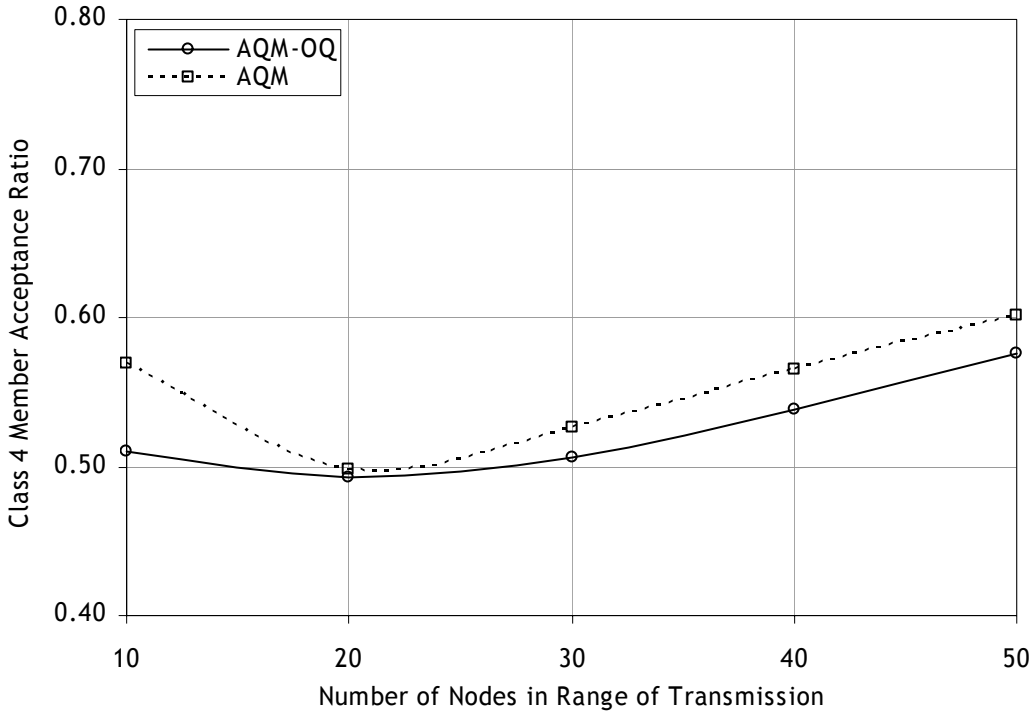



Figure 7.10. Class 4 member acceptance ratio as a function of network density

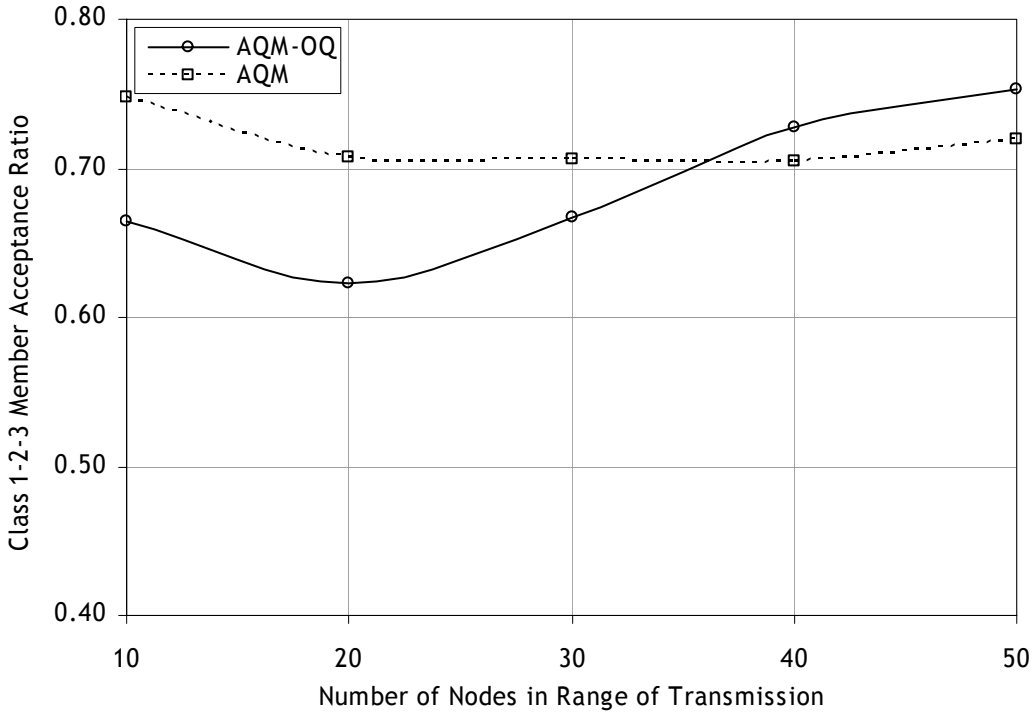



Figure 7.11. Class 1-2-3 member acceptance ratio as a function of network density

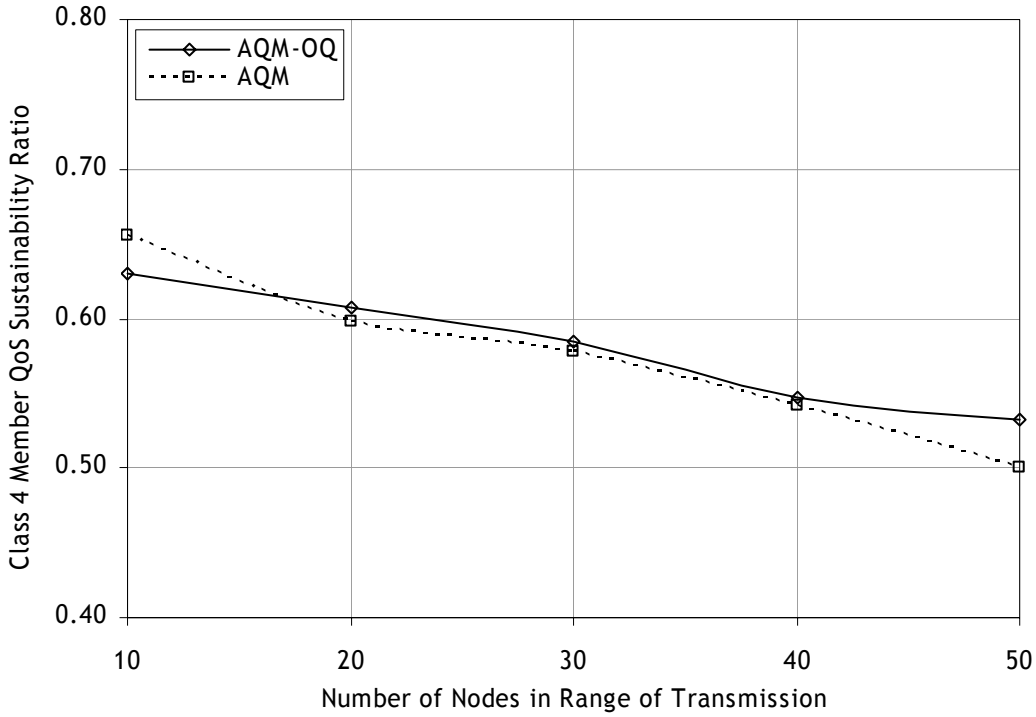



Figure 7.12. Class 4 member QoS sustainability as a function of network density

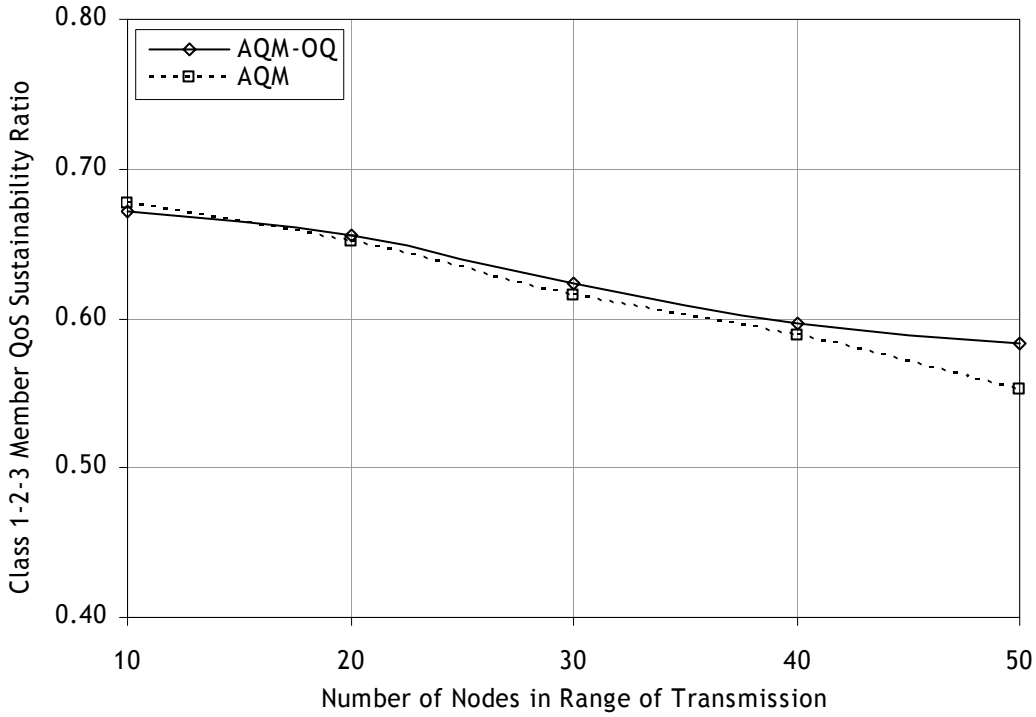



Figure 7.13. Class 1-2-3 member QoS sustainability as a function of network density

The principal behind the objection query mechanism is a more stringent resource reservation scheme to protect nodes from overload. Therefore, a lower member acceptance ratio is expected. Instead, this ratio is actually higher in AQM-OQ than in AQM as the network density increases. AQM-OQ achieves this result by sacrificing more class-four join requests and accepting more requests from the remaining classes in return. As the QoS class definitions in Table 5.1 show, class-four sessions are the most resource-demanding applications in terms of bandwidth. Therefore, AQM-OQ works especially against those applications which belong to the fourth QoS class. However, the rejection of a few more class-four members enables significantly more members from the other three classes with the help of higher network connectivity. As shown in Figure 7.10, the acceptance ratio of class-four session join requests is lower in AQM-OQ, whereas the acceptance ratio of the other three classes is higher than AQM, which can be seen in Figure 7.11. As a consequence of this slight change in the balance of the QoS classes, which is caused by the objection query mechanism, the member QoS sustainability increases for both groups. The results, which also resemble those in Figure 7.8, are shown in Figure 7.12 and Figure 7.13.

### 7.4.2. Overloaded Sessions and Members

Figure 7.14 presents the difference between the member overload ratios of the original and improved versions of AQM. It can be seen that AQM already has a low member overload ratio, which increases only after the network exceeds a certain level of density. Nevertheless, even a small number of session members allocating more resources than available can lead to QoS degradation, if the network density continues to grow or bandwidth requirements become higher. Thus, the lower member overload ratio achieved by AQM-OQ is still important in the sense that it enables the protocol to maintain acceptable QoS conditions for more session members.

Figure 7.15 compares the session overload ratio of AQM-OQ to that of AQM with respect to network density. An overloaded node in the ad hoc network causes all sessions that forward their multicast data packets through that node to become overloaded. Thus, the ratio of overloaded sessions in plain AQM can be disturbing even though it is small. It can be seen that the objection query mechanism improves the performance of AQM by maintaining a lower session overload even under conditions of high network density.

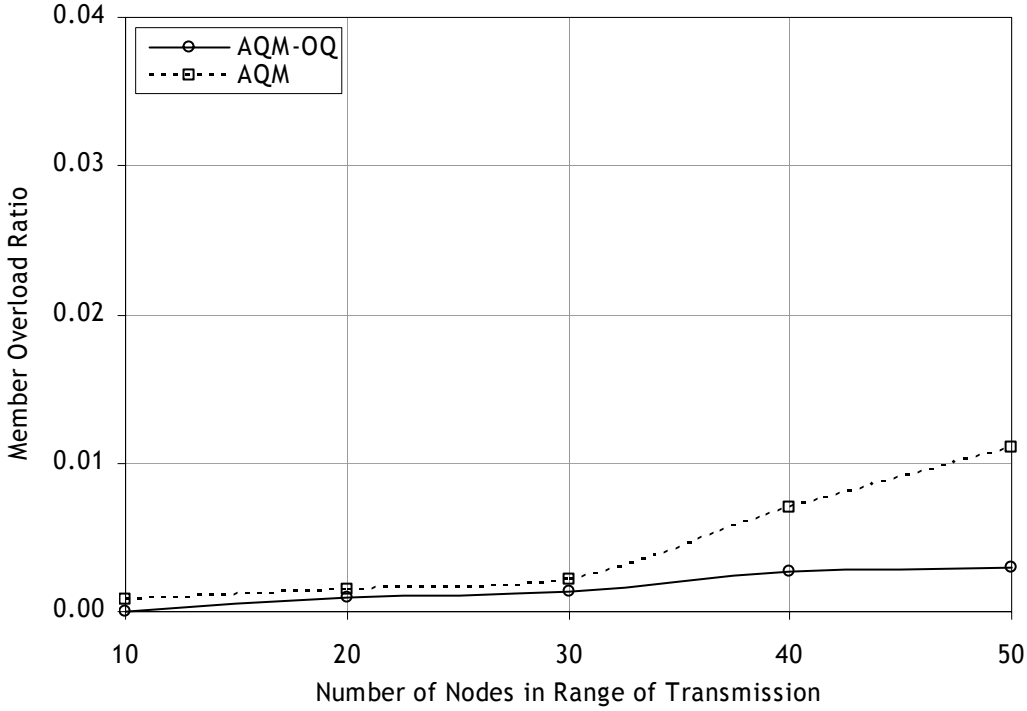



Figure 7.14. Member overload ratio as a function of network density

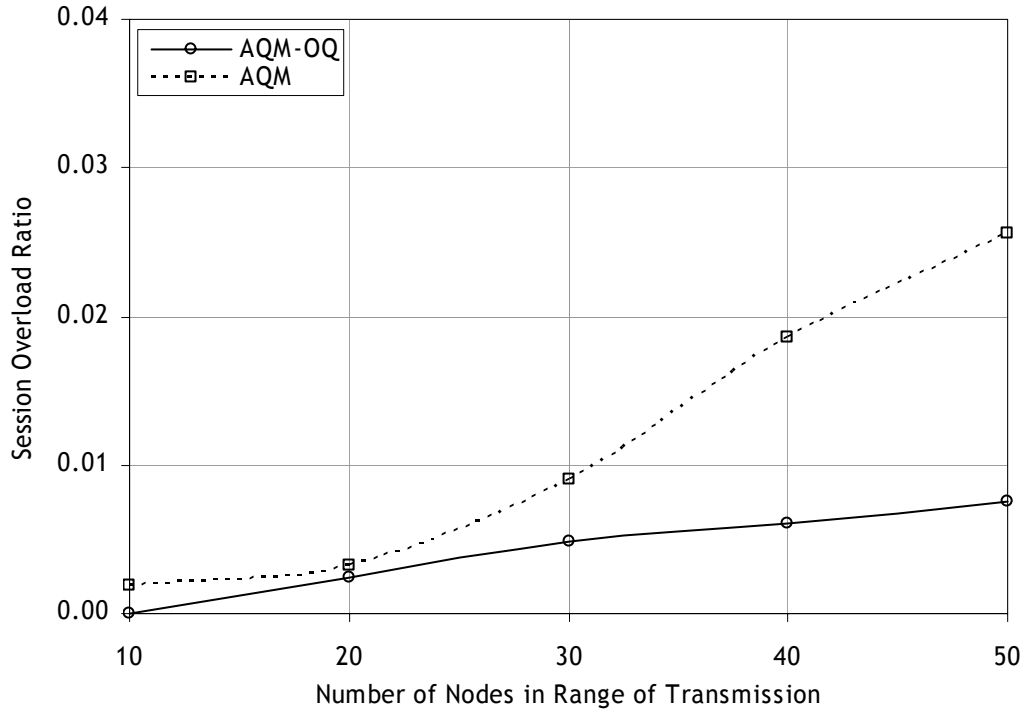



Figure 7.15. Session overload ratio as a function of network density

As mentioned previously, the session overload ratio is an indication of the number of sessions experiencing QoS violations caused by their members due to resource allocations exceeding the network limitations. It is an important criterion since an overloaded session member has a negative effect on all the downstream members of that session, as well as on the downstream members of all the other sessions that are being forwarded simultaneously by the same overloaded member. Thus, AQM-OQ favours the overall satisfaction of sessions over the satisfaction of individual nodes.

AQM-OQ decreases the number of overloaded members and sessions to increase QoS sustainability. It achieves this by inquiring the network about resource availability even for the nodes that are outside the transmission range but still affected by the resource allocations of each other. Other tests are run to see the effect of the transmission range and the node population separately on the performance of AQM-OQ and similar results are obtained. The reader may refer to [106] for the details of these evaluations.

### 7.4.3. Effects on Network Load

Figure 7.16 displays the difference in the ratio of successful background activities between AQM and its improved version. The change of trend in the ad hoc network traffic towards slightly fewer class-four session memberships with the introduction of objection queries enables more best-effort data packets to be delivered successfully between neighbours, since they find better chances of transmission when there are fewer large-size class-four multicast data packets. Thus, in addition to a better overall QoS sustainability, AQM-OQ also achieves a better background traffic efficiency.

Figure 7.17 compares the member control overhead incurred by AQM and AQM-OQ. As expected, AQM-OQ brings a certain amount of additional overhead to the system with the introduction of the objection query messages, which are issued once by each replying node prior to sending the reply and answered by several nodes in case of an overload in their neighbourhoods. However, the overhead difference between AQM and its improved version is not significant. The main reason for this result is that AQM already takes a number of measures to prevent infeasible requests. Therefore, only a small number of nodes need to send objections in response to the queries.




Figure 7.16. Background traffic efficiency as a function of network density

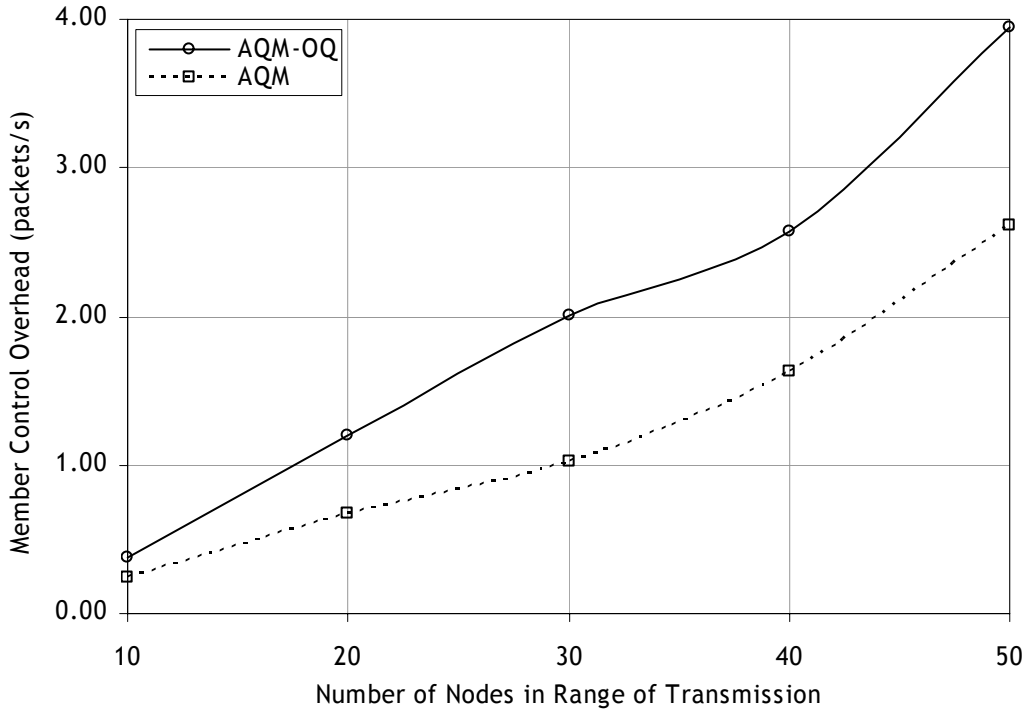



Figure 7.17. Member control overhead as a function of network density

## 7.5. Final Remarks

AQM-OQ, the improved version of AQM, achieves better performance in overload avoidance both at member as well as session levels with an acceptable cost. The increase in control overhead is not very significant and is paid-off by a higher QoS sustainability ratio and background traffic success rate. Even the member acceptance ratios are improved for the less resource-demanding QoS classes. Moreover, the objection query mechanism is an extension to AQM, which is developed for networks that put special emphasis on QoS restrictions. It can be turned off or applied with adaptive conditions for other networks which require higher acceptance rates or lower overhead. An example for such a change is a heuristic rule which says that a node may object to an allocation only if it exceeds the available bandwidth by a certain percentage.

An important point regarding the interpretation of the results achieved by AQM-OQ is that although there are almost no overloaded members and sessions in the network, which means that resource allocation is handled successfully by AQM, it has not been possible to improve the member QoS sustainability beyond a certain level, mainly due to the high data loss rate of the wireless medium. Thus, a QoS-aware multicast protocol can coordinate the usage of the allocated resources more efficiently if QoS is also supported by the lower layers.

It can be argued that the success of an ad hoc QoS multicast routing protocol such as AQM also relies on its ability to adapt to the changes in the network and the preferences of its users. The evaluation of such an adaptive system may require the definition of new performance criteria derived from a weighted sum of the two metrics presented above. The reader may refer to [107] for a detailed evaluation of AQM with two examples of such new performance criteria. Possible combinations of the solutions presented throughout this thesis can be applied as alternate multicast strategies. The weights of the performance criteria can be given to the mobile nodes when they enter the network. This way, AQM can operate in an intelligent manner in order to reach an optimal state with regard to changing network conditions, which leads to a sustainable ad hoc multimedia network with QoS provision.

# 8. AN EXTENSION FOR SERVICE DIFFERENTIATION: PRIORITY QUEUES

As mentioned in Section 1.1, a QoS system is the integration of several important components, such as admission control, resource allocation, congestion control and service differentiation [11, 16-18]. So far, AQM has addressed the first two of these components directly as a QoS-aware multicast routing protocol. It announces its nodes' observations on the existing network status by means of session initiation, update and loss as well as neighbour greeting messages. It distributes information on new multicast sessions only to the extent of QoS provision. It restricts admission to sessions in order not to cause degradation in the perceived QoS of existing session members. It allocates resources to accepted sessions and does not let members accept new connections if residual resource availability is not enough. Once a route is selected for a specific connection request and reservation of resources is completed, these resources are not available to subsequent requests until the end of the granted connection.

Another important component of a QoS system is the congestion control scheme. Congestion is the result of data flow exceeding the capacity of the network. It causes high delay and loss rates. AQM continuously monitors the QoS conditions as perceived by the session members during data delivery. Several metrics such as the delay of and the interarrival time between data packets and the ratio of lost packets to correctly received packets are defined at each node in order to decide on the sustainability of the required QoS for a session. If such degradation of QoS performance is observed, countermeasures can be taken such as reducing the transmission rate, dropping selected packets or rerouting the session to an alternate path. Currently, AQM simply drops the members experiencing unacceptable QoS conditions for a time period longer than bearable to prevent the waste of resources and maintain the required QoS level for other session members. This behaviour can be justified by the general trend in the ad hoc QoS literature towards assuming that it is almost impossible to guarantee the QoS requirements for the entire session [11]. However, other congestion control schemes can easily be implemented by AQM to take action against network congestion in a more sophisticated manner.

The last important component mentioned above is the differentiation between various supported QoS classes by the ad hoc network. There are basically two QoS techniques related to service differentiation, which are priority assignment and fair scheduling. These techniques also represent two distinct levels of implementation. Priority assignment mechanisms are also referred to as packet-level scheduling systems [16]. There is a priority queue in each node that determines which packet to send during the next transmission period based on the service class that the packet belongs to [17]. Thus, packet priorities are set by the respective initial sender of the packet and each node on the path between the source and the destination processes packets according to their priorities and the queuing discipline of the network but independent from the other nodes sharing the same wireless medium as the current node. Fair scheduling algorithms, on the other hand, are also known as node-level scheduling systems, which require the cooperation of all the nodes within a neighbourhood to determine which nodes have channel access priority [16]. They partition resources among flows in proportion to a given weight and regulate the waiting times for fairness among traffic classes [17]. In other words, there is a fundamental separation between the decision about the time when to send the next packet and the decision about the particular packet to send next. The first decision should be made by the node-level scheduler, whereas the second decision concerns the service classes and should be made by the packet-level priority mechanism. Essentially, the approach is to limit the scope of the scheduler to determine only which flow is allocated the channel next and let each flow make its own decision about which packet in the flow it wishes to transmit [108].

In this section, an extension to AQM is presented to address the service differentiation aspect of ad hoc QoS multicast routing. The extension involves the implementation of a priority queuing mechanism at each node which sorts data packets according to their service classes. The aim of such a mechanism is to provide multicast sessions of more stringent delay requirements with quick access to the wireless medium. In the current version of AQM, a delay-sensitive packet does not have any access priority over a more delay-tolerant packet, which reduces the probability of timely delivery for the former. Similarly, such a packet in an overloaded first-in-first-out (FIFO) queue has to wait the same time as other packets. Thus, the objective of priority-based queuing is to differentiate service classes and offer appropriate waiting times to each class [109].

## 8.1. Background and Motivation

A complete system consisting of all the required QoS components is defined as a QoS framework, which provides its users with the negotiated services. Its key design issue is whether it should serve users on a per session or a per class basis. Two of the models which represent these two approaches' counterparts in conventional wired networks are the integrated services (IntServ) model [110] and the differentiated services (DiffServ) model [111], respectively. IntServ routers maintain the state information of sessions and offer various services. They use the resource reservation protocol (RSVP) [112] to reserve resources along a route. However, IntServ is not a scalable model for the Internet. DiffServ aggregates sessions into a limited number of classes and overcomes the scalability problem. Nevertheless, these models cannot be applied directly to mobile ad hoc networks because of the dynamic network topology, limited resource availability and error-prone shared wireless medium. Thus, new QoS frameworks are proposed for ad hoc networks.

Packet prioritization and scheduling, which is an essential means of service differentiation, is also an important feature of these frameworks. For instance, the in-band signalling QoS framework (INSIGNIA) is developed to provide adaptive real-time services in ad hoc networks. It uses a weighted round-robin packet scheduling module [113]. Another model called stateless wireless ad hoc networks (SWAN) facilitates feedback-based control mechanisms to support real-time services and service differentiation [114]. It uses a packet classifier to differentiate between packets and a traffic shaper to regulate best-effort traffic as well as to prioritize real-time packets. A third model, namely the proactive real-time medium access control (PRTMAC) protocol, is a cross-layer framework to provide real-time traffic support and service differentiation to highly mobile ad hoc networks [11]. It defines three priority classes, whereby lower-priority connections may be preempted to support a high-priority class. The QoS Protocol for Ad Hoc Real-Time Traffic (QPART) is another cross-layer framework, which introduces the concept of virtual stations [115]. Each node maintains a queue for each traffic class. Each queue tries to access the channel independently. A scheduler resolves virtual collisions. Thus, each queue can be modelled as a separate station. Priorities are assigned by using the minimum and maximum contention windows of individual classes and the management is left to the underlying MAC protocol.

Schedulers can be broadly classified as work-conserving and non-work-conserving [116]. A work-conserving scheduler does not remain idle if there is a packet in its transmission queue. In contrast, a non-work-conserving scheduler may become idle even though there is a backlogged packet in the queue if it is expecting a higher-priority packet to arrive. As a result, non-work-conserving schedulers generally have higher average packet delays. They are useful in applications where delay variance is more important than delay. A scheduler can be designed to serve packets based on their timestamp values. The packets are placed in their respective queues and sorted in accordance with their timestamps. Round-robin schedulers do not use timestamps and can be more easily implemented. However, timestamped schedulers can provide better QoS guarantees. Finally, in sorted-priority schedulers, each session has a different priority level and packets are transmitted according to their session priorities.

Today, QoS and best-effort traffic coexist in most real-world networks, including conventional wired as well as wireless networks. Therefore, it is logical to assume that they will also coexist in mobile ad hoc networks. In such an environment, it is necessary to implement a scheduling mechanism in order to satisfy the requirements of both traffic types and achieve resource efficiency. Generally, the QoS traffic is not affected by the best-effort traffic if the resource reservation issue is taken care of. However, best-effort traffic may suffer if the overall traffic flow is not regulated. To provide fairness for the best-effort traffic, hierarchical packet fair queuing (H-PFQ) algorithms are designed, which have the potential to simultaneously support guaranteed real-time, rate-adaptive best-effort and controlled link-sharing services [117]. However, these algorithms work only for a fixed set of flows, which makes them inappropriate for mobile ad hoc networks, where new QoS flows join or existing QoS flows leave the network all the time. Thus, their decentralized versions are developed, which are hierarchical scheduling algorithms that adjust the bandwidth allocated to the flows in such a way that every QoS flow always receives a minimum bandwidth guaranteeing the required QoS and the residual bandwidth is shared fairly among all best-effort and QoS flows. To achieve these goals, the link capacity is divided between two logical servers on the first level and then the respective flows are scheduled by these servers on a second level. Both servers are implemented by a modified version of weighted fair queuing (WFQ) algorithm, which also eliminates the scheduling interference between the two flows [118].

### 8.2. Priority Queuing Based on QoS Classes

In order to provide service differentiation to various classes of multimedia applications and meet their QoS requirements more effectively, a priority queuing mechanism based on service classes is integrated into AQM. A priority module is added to the node structure, which is described previously in Section 3. Consequently, the initial protocol architecture depicted in Figure 3.1 is revised such that it includes this new module as shown in Figure 8.1. The details of the other four modules are omitted since they remain basically unchanged.

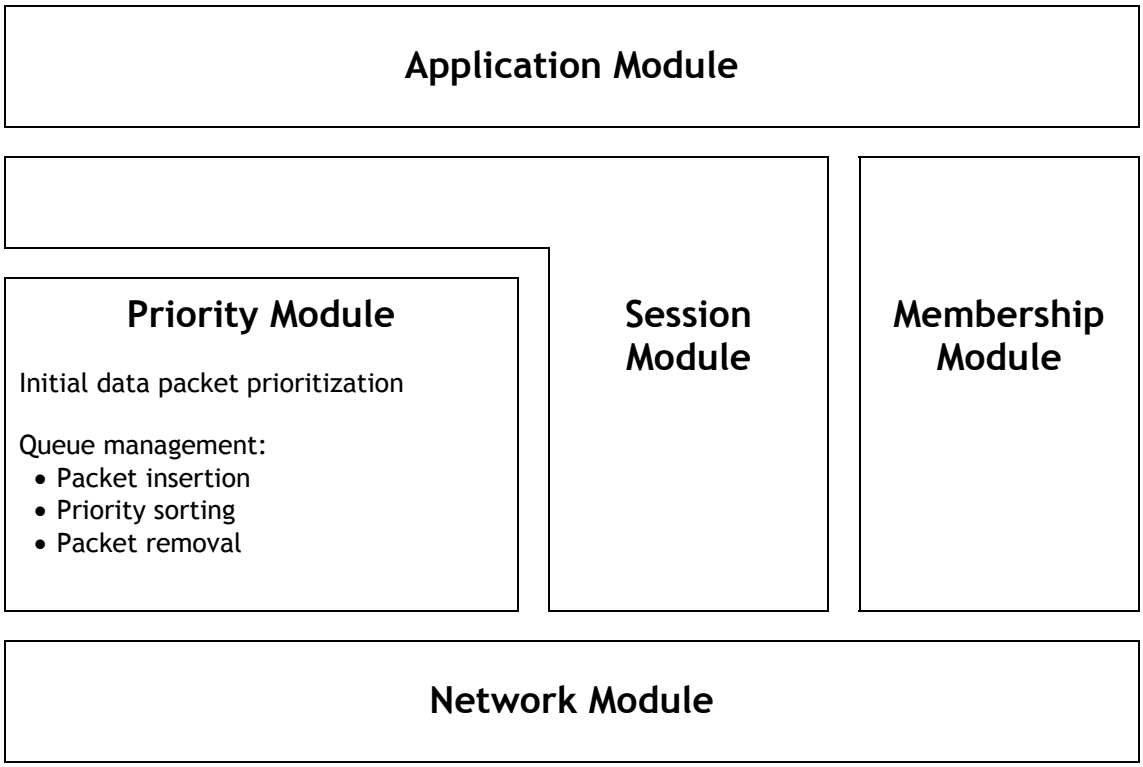



Figure 8.1. Extended architecture of the AQM protocol

Data packets may arrive at the priority module only from the session module and depart from the priority module only to be delivered to the network module. Thus, the priority module puts arriving packets to a transmission queue in accordance with their priorities. On the other hand, it is also the task of this module to select a packet from the queue and deliver it to the network module to be transmitted. If the node is the server of a session, the priority module has to set the priorities of the individual data packets originated from its own application layer first. If the node is a forwarder in a session, the

priority of the incoming data packet is already set by its server. A node can be the server of a session and a forwarder in several other sessions at the same time. The fact that a node deals with a number of sessions simultaneously is the main reason why it has to consider their service classes and prioritize data packets. Actually, it is the selection of the right data packet to remove from the queue for transmission that provides service differentiation which is the main objective of the priority module.

There are a number of priority queuing algorithms in the literature that can be implemented by the new priority module of AQM, some of which are summarized in Section 8.1. One of the relatively easy implementations requires a single priority queue which data packets are inserted into according to their service class priorities and arrival times. In this section, AQM adopts this simple approach. Thus, each data packet is queued for transmission in order of its service class and timestamp. Here, the starvation of packets of lower-priority can be a problematic issue. However, the design of the priority module is independent from the other modules. Thus, more sophisticated queuing algorithms can be chosen to work within the AQM protocol architecture. For instance, the priority module can maintain a separate queue for each session that it is involved in and ensures that each queue is served for a period of time which is fairly proportional to the bandwidth allocation ratio of the corresponding session. This alternate approach closely resembles the WFQ model, which is a fair scheduling algorithm and not a priority queuing one. Since bandwidth allocation is an issue handled by the session and membership modules of AQM, the priority module developed in this section is content with the single queue approach where application classes are used to differentiate between data packets.

An important part in the design of the priority module is the method it adopts to trigger the process for the selection of the next packet that should be delivered to the network module. As mentioned in Section 8.1, a scheduler can be work-conserving or non-work-conserving. Similarly, the priority module can utilize a passive or an active queue. In the former case, the priority module is polled by another module for the next data packet to be sent. According to the architecture of AQM, this other module should be the network module, which also has to check the underlying MAC layer to see if it is ready for the transmission of the next packet. If it is assumed that the MAC layer maintains its own transmission buffer for the segmentation of large packets, where packets are processed in a

FIFO order, this buffer should not be empty as long as there are packets to be transmitted for the sake of protocol efficiency. On the other hand, data packets should primarily queue up within the priority module for a dynamic and efficient prioritization to be possible. From AQM's point of view, these two requirements yield a result which is that ideally there should be at most two packets at the transmitting end of the MAC layer, one being processed and the other waiting in the transmission buffer. However, this method involves extra coordination with the MAC layer and eventually leads to a cross-layer design, which is beyond the scope of the design objectives of AQM.

In order to overcome the coordination problem confronted, the priority module can actively trigger the packet selection process. In other words, it decides when to remove the next packet from the queue and send it to the network module by itself. This is a much simpler method if the packet processing interval of the priority module is set realistically. The module should process packets fast enough such that it does experience packet overflow. On the other hand, it should be slow enough to allow data packets to be queued and prioritized. Thus, the priority module schedules the delivery of the next data packet by using the packet size of the last data packet that has been sent and the bandwidth of the wireless medium, such that the MAC layer can complete the transmission of that packet and get ready for the next one in time.

The last important issue to be mentioned is the treatment of control packets. Since these are packets of highest priority, they do not enter the same queue as data packets at all. Instead, they are delivered by the session and membership modules directly to the network module, by-passing the priority module and obtaining priority over data packets.

In summary, the priority module of AQM employs a single priority queue, where newly arrived data packets are sorted in accordance with their service classes and their order of arrival. The data packet with the highest priority is the one belonging to the premium service class and the earliest arrival time in the queue. This is the next packet to be sent to the network module for transmission. The priority module does not remain idle unless there are no packets to be processed. Thus, it acts as an entity responsible only for the arrangement of data packets in such a way that the packet with the highest priority is delivered to the MAC layer first.

## 8.3. Performance Evaluation

The results achieved by this extension to AQM with a data packet priority queuing mechanism for service differentiation are evaluated in this section by examining the delay performance of each individual service class as well as that of the background traffic. For this purpose, the set of applications defined in Table 5.1 are prioritized according to their packet interarrival time and delay requirements. Thus, applications of class one are given the first priority, whereas class-two applications the second, class-three applications the third and finally, applications of class four are given the fourth priority.

The simulations are conducted using OPNET Modeler 11.5 Educational Version and Wireless Module [100]. The tests are repeated 10 times for each data point and results are aggregated with a 95 per cent confidence interval. AQM nodes are modelled by the existing application, session, membership and network managers with the new priority manager added to the node architecture as shown in Figure 8.1. The average session durations of the applications, which are also given in Table 5.1, are halved in order to achieve similar membership dynamics with more condensed simulations of 15 minutes' time. The multicast inactivity period is decreased to 60 seconds, whereas the background traffic inactivity period is set to 100 seconds. The parameters of the mobility model and other simulation settings remain the same as in Table 5.2.

In order to eliminate possible effects of the simulation environment on the results due to additional processing delay incurred, the AQM priority queue is compared with a FIFO queue which is also implemented on AQM. Thus, the delay performance of AQM is evaluated with and without service differentiation where all other conditions are the same.

Two sets of simulations are conducted with these common parameters. The first set examines the effect of network density on AQM's priority mechanism. In this set, the average number of neighbours within a node's transmission range is used to represent the density. The second set, whereby the ratio of sessions which belong to the heaviest service class is the variable, aims to test the effect of the distribution of QoS classes on service differentiation in AQM. In this set, the classes other than the heaviest share the remaining occurrence ratio equally. The details of the simulation variables are given in Table 8.1.

Table 8.1. Simulation variables of the queuing performance evaluation

| Set | Variable Description | Number of Nodes in Range | Heavy Multicast Class Ratio | Background Data File Size |
|---|---|---|---|---|
| 1 | Network density | 20 – 40 | 0.25 | 2 MB |
| 2 | QoS class distribution | 20 | 0.10 – 0.90 | 2 MB |

### 8.3.1. Effect of Network Density

Figure 8.2 through Figure 8.4 show the difference achieved in multicast data packet delivery delay of the three QoS classes with higher priorities. By utilizing the class-based priority queuing mechanism, each AQM node arranges the transmission order of the packets in favour of those with stricter QoS requirements in terms of delay. However, the difference is not particularly high. In addition, it can also be seen that there is a trend of growing delay for all three classes as the network density increases, even though the average delays of the priority queue are still lower than those of a FIFO queue.

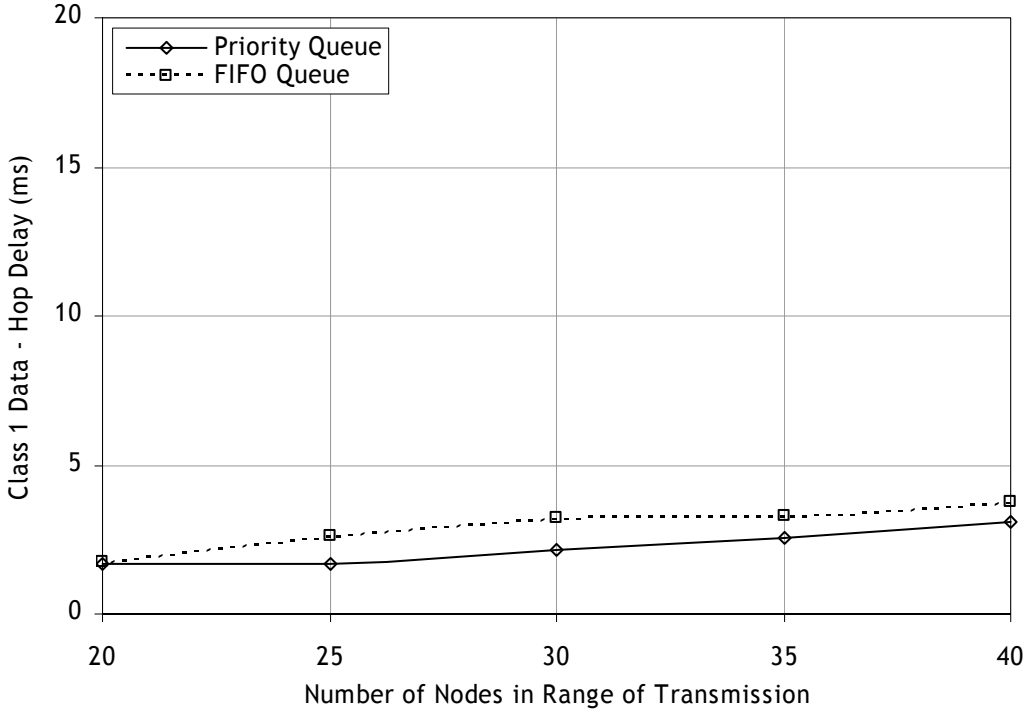



Figure 8.2. Class 1 data packet delay as a function of network density

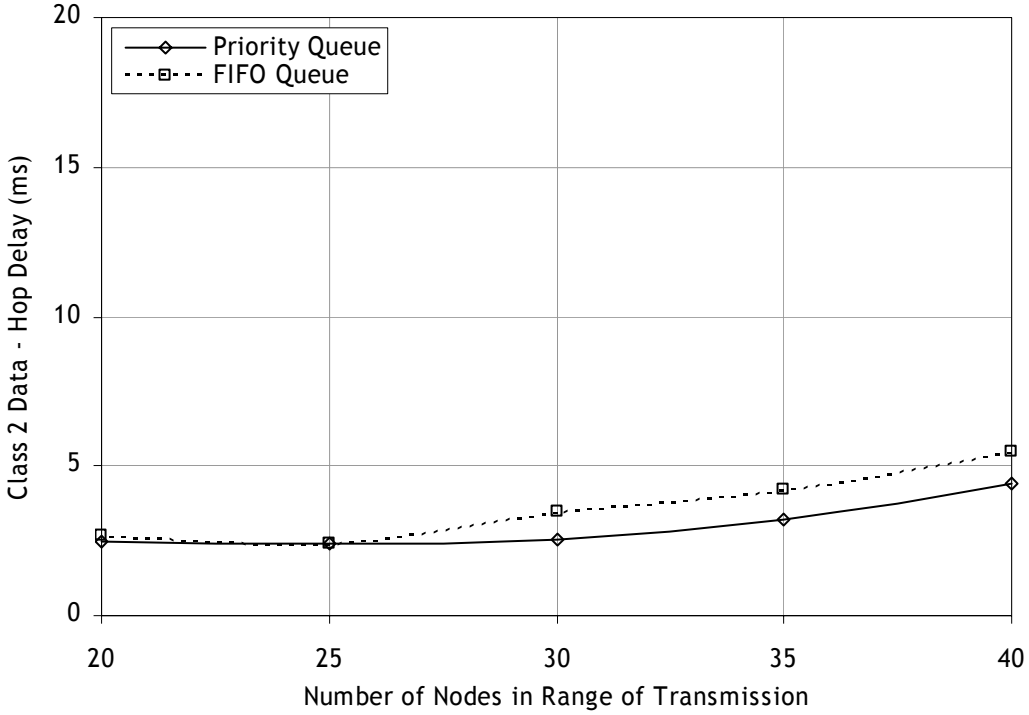



Figure 8.3. Class 2 data packet delay as a function of network density

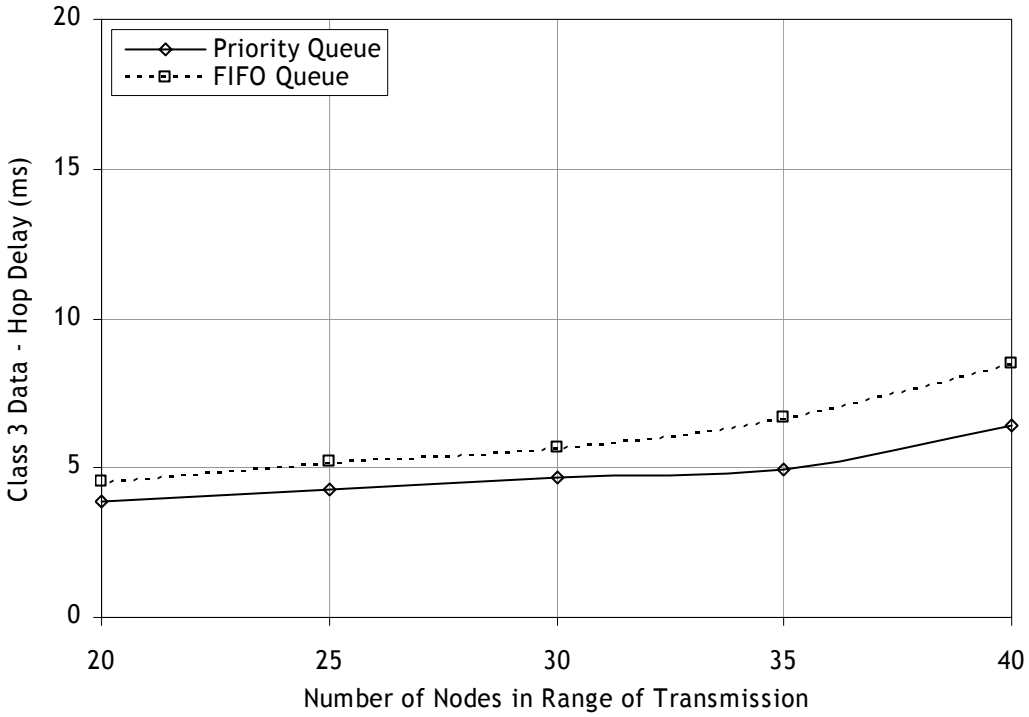



Figure 8.4. Class 3 data packet delay as a function of network density

The basic reasoning for both of these observations is that all nodes have to contend against their neighbours to access the wireless medium first, since there is no QoS-related coordination between them. In other words, since channel access is not scheduled based on packet priorities, each node within a neighbourhood has an equal chance to capture the wireless medium regardless of the class or priority of the packet it is about to transmit. It is very time-consuming when e.g. a class-one packet selected by its node to be transmitted has to wait for another node's transmission of a large-size class-four packet since the nodes do not have any information on the relative priorities of their packets. As the network density grows, the probability of confronting with such unfair situations increases for each node. Thus, the effect of priority queuing is limited.

Figure 8.5 displays the class-four delay results of the priority and FIFO queues with regard to growing network density. Since the priority queue forces the data packets of this class to wait for the other three classes, the averages of the priority queue are higher than those of the FIFO queue. Moreover, the absolute value of the delay is also large due to the time required to transmit a large-size class-four data packet. The trend of increased delay can be observed for this class as the network density grows.

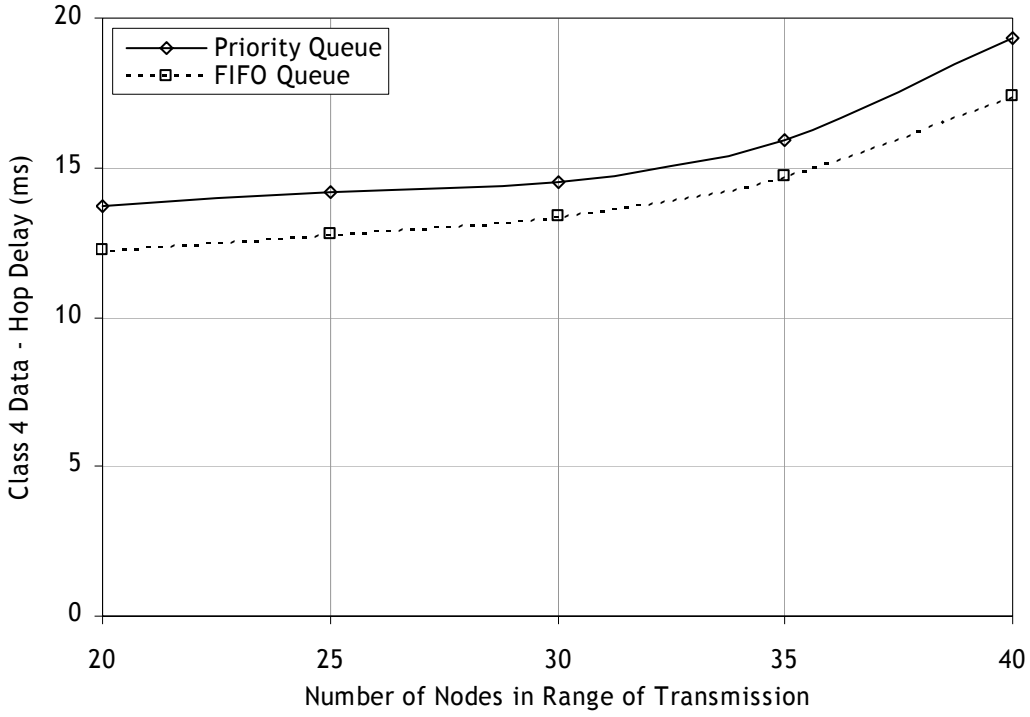



Figure 8.5. Class 4 data packet delay as a function of network density

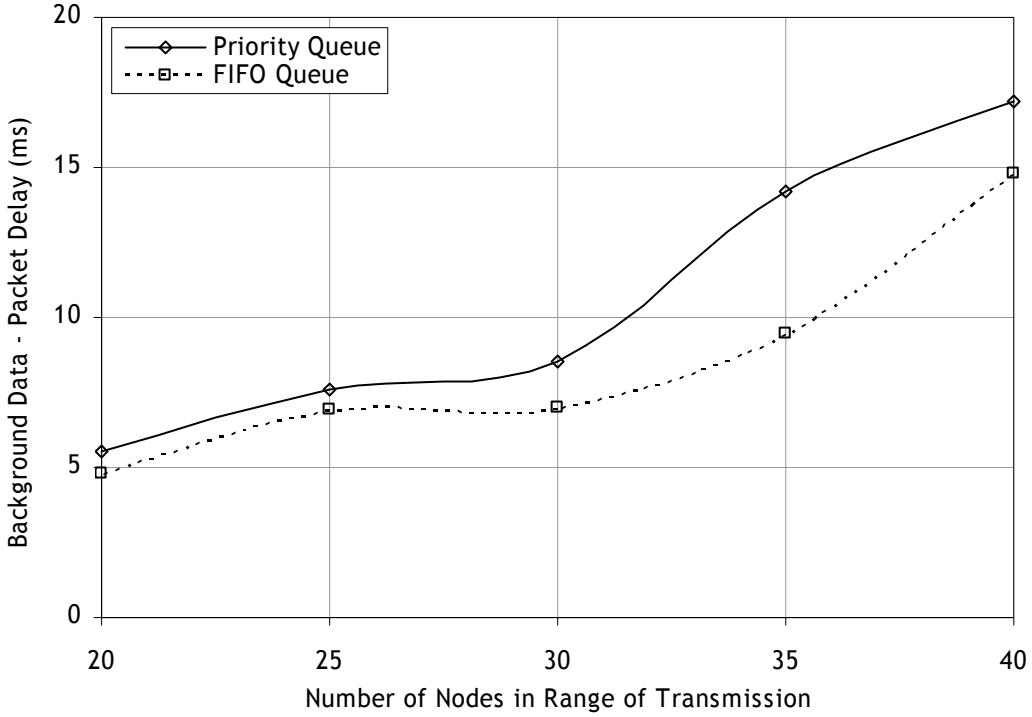



Figure 8.6. Background data packet delay as a function of network density

Finally, Figure 8.6 presents the packet delay experienced by the background traffic under changing network density conditions. AQM handles these packets with the least priority, since they belong to a best-effort service which is error-sensitive but tolerant of delay. However, the absolute value of delay is still less than that of class-four packets due to their small packet size. On the other hand, background traffic packets are also the most affected by the increase in the number of neighbours, since they have to wait for all other packets in addition to very high contention periods.

### 8.3.2. Effect of QoS Class Distribution

Figure 8.7 through Figure 8.9 show the improvement in packet delays achieved by the three higher-priority classes through the priority queue. Similar to the results presented in the previous section, the delays these classes experience are slightly lower when they are prioritized over the fourth class. However, the delays grow as the ratio of the class-four applications increases, mainly due to the increase in the large-size packets being transmitted in the network. Even though packets are prioritized within a node, contending nodes within its neighbourhood inhibit the queuing discipline from being more effective.

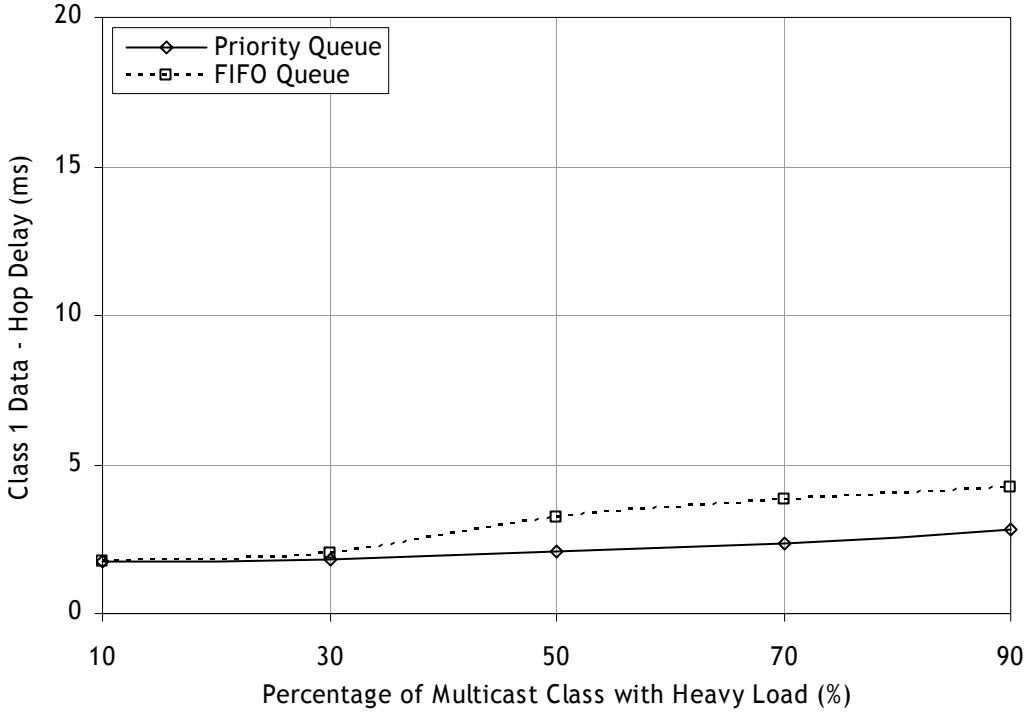



Figure 8.7. Class 1 data packet delay as a function of QoS class distribution

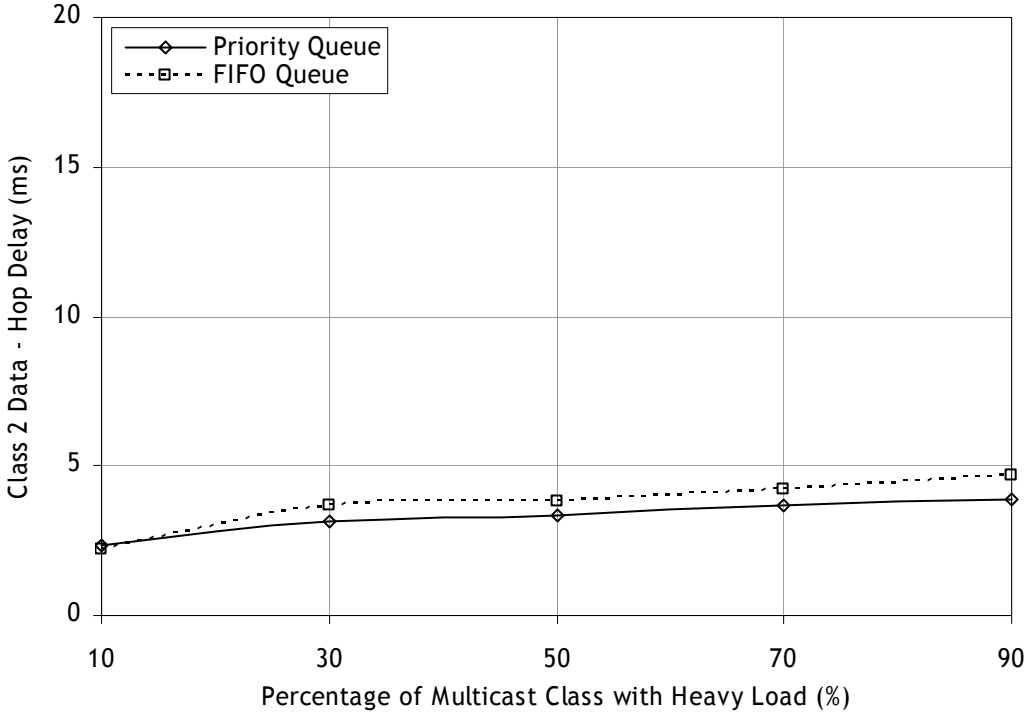



Figure 8.8. Class 2 data packet delay as a function of QoS class distribution

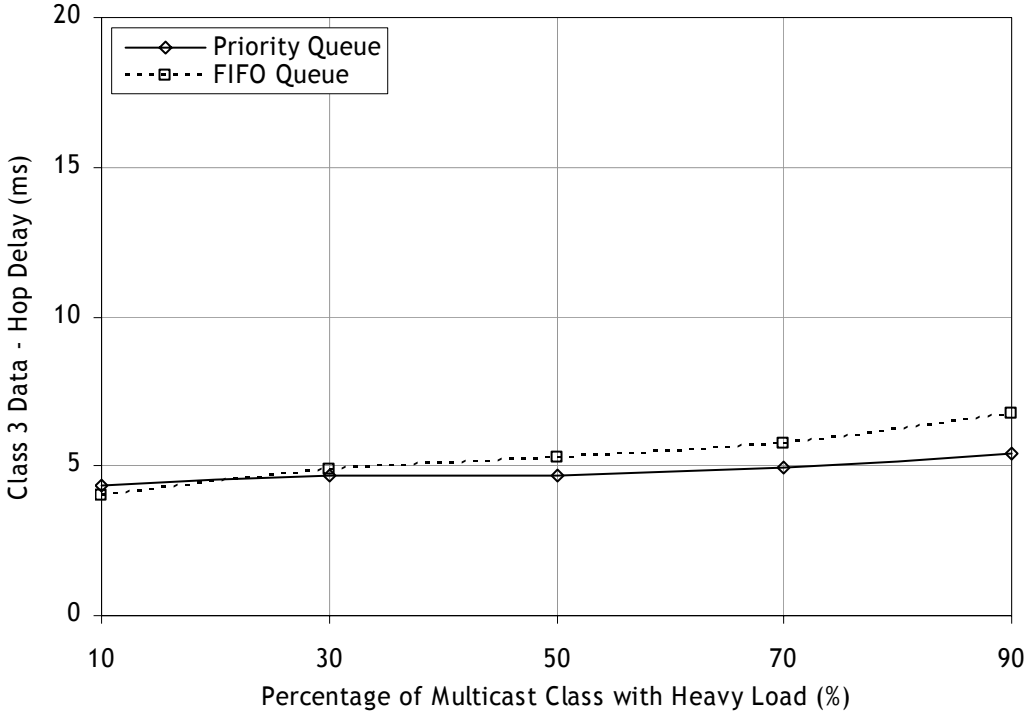



Figure 8.9. Class 3 data packet delay as a function of QoS class distribution

An interesting phenomenon revealed by Figure 8.9 is that class-three applications experience slightly lower delay with the FIFO queue until class-four applications reach a certain percentage. There are a significantly smaller number of class-four packets in the network when the ratio of heavy multicast traffic is low. This enables the FIFO queue to process class-three packets quicker in the absence of the large-size class-four packets. Thus, the FIFO queue functions in favour of class-three packets when there are very few class-four packets in the queue, due to the large size of the latter. On the other hand, class-three packets have practically the least priority for the priority queue as long as there are not many class-four packets. When the ratio of class-four applications increases, however, class-three applications start being served better by the priority queue.

Figure 8.10 displays the change in the delay performance of the class-four packets. As the ratio of sessions belonging to the fourth class increases, the growing number of large-size data packets also affects the other packets from class-four. Thus, the delay increases in a similar manner as the other classes. In addition, the class-four packets processed by the FIFO discipline experience a lower delay since they are served in order of arrival and get a better chance of being transmitted earlier.

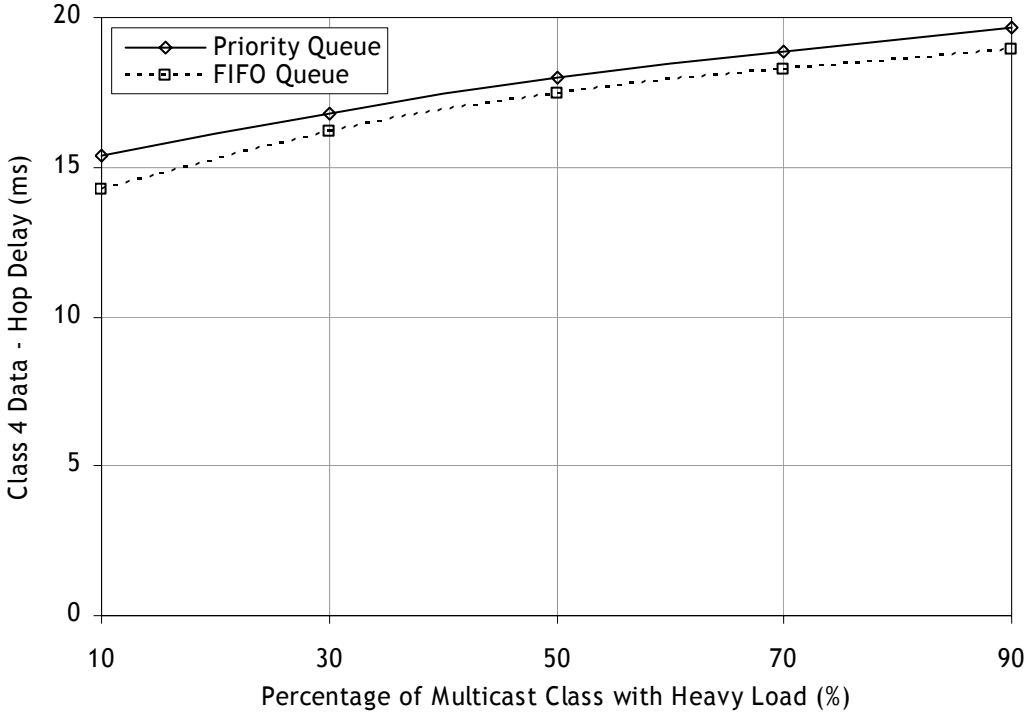



Figure 8.10. Class 4 data packet delay as a function of QoS class distribution

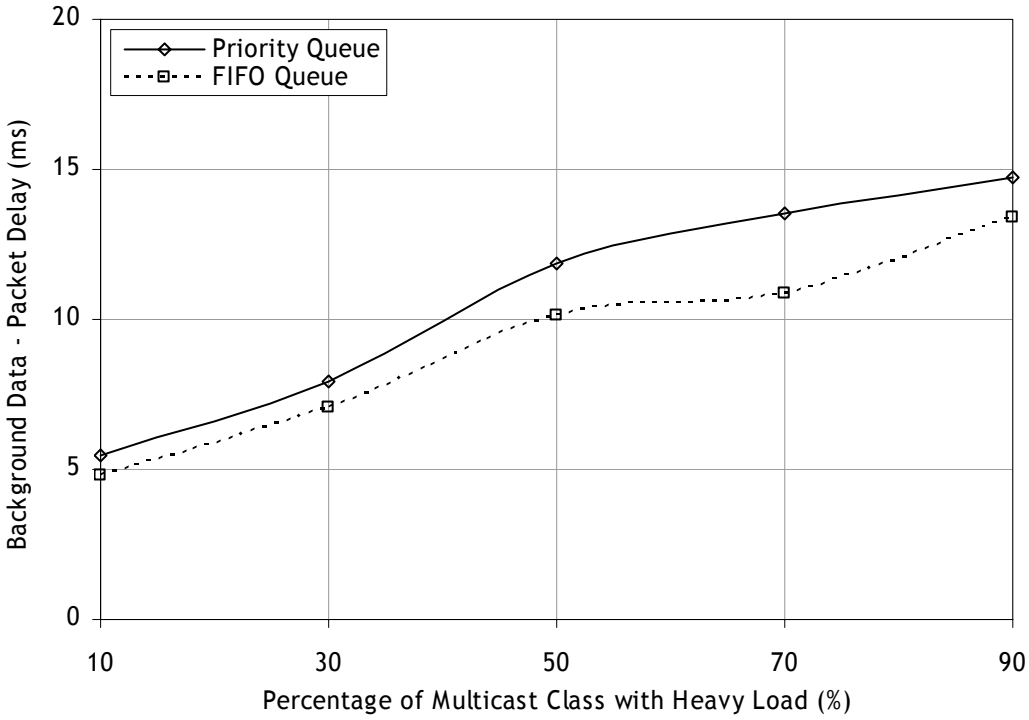



Figure 8.11. Background data packet delay as a function of QoS class distribution

Finally, the packet delay of the background traffic, which has the lowest priority in the AQM priority queue, is presented in Figure 8.11. Since they are served even after the class-four data packets, their delay values are also higher than when they are served in a FIFO order. The delay increases with the ratio of class-four applications, since the fourth class represents heavy multicast traffic with the largest packet sizes.

## 8.4. Final Remarks

In this section, AQM is enhanced with a priority queuing mechanism in order to integrate service differentiation into the protocol architecture. In a mobile ad hoc network where multicast applications that belong to various QoS classes may coexist, AQM nodes are able to differentiate between data flows according to packet priorities and arrange the transmission order of packets such that sessions with more stringent interarrival time and delay requirements are served before the others. Further measurements are made in order to evaluate the effect of priority queuing on AQM with regard to some of the other performance metrics presented in Section 5.1, such as end-to-end delay, multicast data loss and background success rates. The end-to-end delay is observed to be proportional to the hop delay, whereas the results of the latter two do not differ significantly for FIFO and priority queues. Thus, the improvement in the hop delay is not sufficient to decrease the number of dropped members and increase QoS sustainability. Since these measurements give similar results, they are not explicitly presented here to avoid repetition.

The priority queue implemented in this section can be improved or AQM can adopt alternate queuing disciplines with more sophisticated sorting and selection algorithms in order to achieve better service differentiation. For instance, the priority module can be designed such that it maintains a certain number of queues in parallel, each of which is responsible for a certain service class and sorts its contents according to the arrival or creation time of data packets. The module inserts arriving packets into their respective queues. When it is time to remove a packet from a queue and send it to the network module for transmission, it selects the queue in an intelligent manner such that starvation of low-priority packets is avoided. This may be the packet of highest priority from the queue where longer waiting times are experienced. There are many other possibilities concerning the implementation of queues in the literature.

Another important issue which may be related to the priority module of AQM is the possibility of traffic monitoring. Since the priority module can observe all data traffic crossing through a node and also be made aware of the resource reservations for each of these flows, it can take action in response to QoS degradation. In the current version of AQM, this is the task of the session module, which drops the node off the session as soon as such degradation is detected in it. Instead, the priority module can drop packets pushing but not yet violating the QoS limits according to a packet priority scheme in case of possible QoS degradation and try to prevent nodes from dropping. However, the benefit of packet dropping is limited since the main reason of QoS degradation experienced by a node is poor packet reception performance, which means that the problem possibly lies in some node up the stream. In this case, upstream session members should be notified of QoS degradation, which leads to the well-known topic of congestion control.

Since the design of AQM does not involve cross-layer approaches, priority scheduling among neighbours is not considered so far. However, without the support of the underlying layers to coordinate access between the nodes in a neighbourhood, it is not possible to distinguish a node about to transmit high-priority data packets from other nodes carrying packets of lower priority. Thus, node-level service differentiation, the performance of which is presented through simulation results in this section, can be much more efficient if it is supported by network-level priority scheduling. However, this is a very broad and important research topic concerning the development of a QoS-aware MAC layer protocol. The MAC protocol determines which node should transmit next when several nodes contend for the wireless channel. Wireless MAC protocols with their basic channel-sensing and random backoff schemes are only suitable for best-effort traffic in mobile ad hoc networks. Supporting real-time traffic in these networks is a very challenging task [11]. There are, however, efforts to provide QoS support at the MAC level. These efforts mainly address two of the important issues regarding QoS, which are resource allocation and service differentiation. There are also various contention-based protocols with scheduling mechanisms, which focus on packet scheduling at the nodes and transmission scheduling of the nodes. Unfortunately, these issues are beyond the scope of the AQM protocol.

# 9. CONCLUSIONS

The expectations of the wireless user shifting towards high quality, group-oriented, mobile multimedia communication affects the needs of today's networks. Novel wireless networking technologies embedded into portable computing devices enable an ever-growing number of users to communicate with each other while on the move, i.e., without being connected to a wired infrastructure. As soon as the user becomes part of such a wireless network, however, a series of heavy administrative tasks have to be accomplished to configure the device. Ad hoc networks are self-organizing communication entities which take over this burden and make the user enjoy full wireless functionality.

The increasing amount of multimedia content shared over various communication media today makes QoS-related, resource-efficient routing strategies very important for ad hoc networks. Moreover, the increasing number of group-oriented applications also necessitates the efficient utilization of network resources. The multicast communication model is a promising technique which can achieve this efficiency by facilitating the inherent broadcast capability of the wireless medium and minimizing bandwidth consumption, packet processing and delivery delay. However, it is not easy to integrate QoS mechanisms into a multicast routing protocol.

The multicast routing protocol presented in this thesis, AQM, provides ad hoc networks with the features mentioned above. It keeps the network up-to-date on the availability of sessions with regard to QoS considerations. It controls the availability of resources throughout the network and ensures that the users of an application do not suffer from QoS degradation due to bandwidth allocations exceeding the limits of the shared wireless medium. In its bandwidth calculations, AQM takes the continuity property of multimedia data into consideration and checks bandwidth availability along a virtual tunnel of nodes. It also facilitates an objection query mechanism to inform nodes of the possible overload of others, which cannot be detected directly. AQM sets limits to path length in terms of hop count and checks them in order to satisfy the delay requirements of multimedia applications. By applying these instruments, AQM is able to eliminate infeasible requests for membership preliminarily at their sources.

AQM introduces several ideas which are novel to multicast routing in mobile ad hoc networks. Most importantly, it integrates QoS-aware strategies into multicast routing for admission control and resource allocation. It defines the bandwidth requirement of a multimedia session as a continuous flow of data and makes more accurate decisions on resource availability. It also provides a utility which allows a node to collect information on resource allocation beyond its neighbours and revise its own reservation decisions, which can otherwise lead to excessive resource usage within a neighbourhood. At each node, there is a priority queue which sorts multicast data packets to be transmitted according to their traffic classes. This queuing discipline helps AQM differentiate between service classes, give higher priority to those applications with more stringent QoS restrictions and fulfil their requirements even though they are significantly harder to meet. Finally, AQM is able to evolve the initial multicast tree into a mesh during data flow to improve robustness. A node connects to the existing multicast graph regularly through a single forwarding member but is allowed to register itself with additional forwarders if it starts receiving multicast data from them. This operation increases robustness since a node does not only depend on a single predecessor for its connection to the multicast session.

Service satisfaction is the primary evaluation criterion for a QoS-related protocol. Therefore, new metrics are defined to compare AQM to a non-QoS protocol, in addition to several regular metrics such as multicast as well as background traffic success rates, end-to-end delay, packet interarrival time, packet loss and the control overhead. One of these new metrics measures the sustainability of QoS conditions at members once they are accepted to a session. In addition, overloaded members and sessions can be observed to examine the efficiency of AQM in resource allocation. AQM is evaluated with regard to these metrics under a realistic network scenario, where multiple QoS classes are supported and there are no restrictions on the number of simultaneous sessions or members. The results give a good insight to the quality of AQM. Simulations show that there are significant performance differences between the two protocols for members and sessions. By applying QoS restrictions to the ad hoc network, AQM achieves lower delay, loss and overload and improves the multicast efficiency for members and sessions. Without a QoS policy, users experience difficulties in getting the service they demand as the traffic load grows and activity rates increase. AQM proves that QoS is not only essential for, but also applicable to multimedia communication in mobile ad hoc networks.

AQM has a simple, flat network structure where all nodes are equal. It avoids complicated network topologies such as hierarchical or clustered structures, which are challenging in terms of design and maintenance and present points of failure. However, it is possible to adapt AQM to a clustered network to scale with network size. Intra-cluster multicast sessions can be handled by AQM, whereas inter-cluster communication can be managed by a higher-layer, hierarchical version of it, still providing the network with QoS features. It is not a realistic assumption that a mobile network can afford a pure on-demand protocol if it has to support QoS. Therefore, AQM proposes a hybrid method in terms of multicast routing with table-driven session management and on-demand verification of QoS information upon the initialization of a join process. It also presents a hybrid graph structure since it starts a multicast session with a tree and develops it into a mesh during the course of the session.

## 9.1. Future Research Directions

AQM is a multicast routing protocol for mobile ad hoc networks, which develops QoS strategies in order to satisfy the service-level requirements of the mobile multimedia user. It is possible to improve AQM further with the help of some additional information collected by each node in a distributed manner and shared among the neighbours in the network. Nodes can measure their queue sizes and estimate their average queuing delays. They can also measure their processing delays and derive a relation to the number of sessions being processed by them. More sophisticated admission, reservation and routing decisions can be made if these observations are shared among neighbours. AQM prevents excessive allocation of bandwidth and helps nodes experience less contention, which also affects delay. However, the queuing and processing delays give a more precise idea about the QoS level of a particular node. Since the propagation delay of the wireless medium is negligible, the current delay is mainly the sum of contention times and transmission delays. Therefore, this additional information can be valuable for AQM to select paths of lower delay.

An interesting future work is the development of an adaptive system which applies alternate AQM strategies, such as enabling or disabling the objection query mechanism, the class-based priority queue of data packet transmission and the tree-to-mesh evolution

decisions, in accordance with general user preferences which can be collected from the mobile nodes when they enter the ad hoc network. Another topic which calls attention is the efficient rerouting of multicast sessions when changes occur in the network topology as a result of mobility or varying QoS conditions. On the other hand, the implementation of a realistic mobility model is also very important for an accurate evaluation of these protocols. Mobility changes the network topology constantly, which has a profound effect on the network characteristics. It is a good idea to evaluate ad hoc network protocols with multiple mobility models. Ad hoc applications with team collaboration and real-time multimedia support necessitate group mobility, which improves performance if protocols take advantage of its features such as multicast routing.

Some of the recent multicast routing protocols in the literature can be assessed using the new performance criteria proposed in this thesis to have an alternate view to their performance in terms of QoS as experienced by the user. Their current evaluations are based on the conventional metrics mentioned above which lack in the QoS perspective.

The scope of this thesis is the design of a QoS-aware protocol for multicast routing in mobile ad hoc networks and to validate it as a feasible and useful one at the higher layers. However, further study is necessary to observe the effects of the underlying data link layer protocols on the performance of AQM and develop the necessary means to improve this performance. For instance, the prioritization of QoS classes is easily realized for a single node, but priority scheduling has to be supported at the neighbourhood level to achieve coordination among nodes and give access priority to those carrying more QoS-sensitive traffic. The reliable delivery of multicast data is also a hard task due to the request-to-send/clear-to-send (RTS/CTS) signalling problem in broadcast routing. These issues and others make the existence of an underlying QoS-aware MAC layer a very important requirement. The MAC layer is also responsible for resource reservation and the acquisition of available link bandwidth information, which is another significant issue involving infrastructure decisions. Therefore, it is necessary to implement a realistic MAC layer and simulate ad hoc network environments closer to real life scenarios. On the other hand, AQM is independent of the design of lower layers and, within the scope of this thesis, efforts have been made to maintain its integrity by addressing some of these issues in higher layers.

# REFERENCES


1. Chlamtac, I., M. Conti and J. J.-N. Liu, "Mobile Ad Hoc Networking: Imperatives and Challenges", *Ad Hoc Networks*, Vol. 1, No. 1, pp. 13-64, July 2003.

2. Obraczka, K. and G. Tsudik, "Multicast Routing Issues in Ad Hoc Networks", *Proceedings of ICUPC '98 – the IEEE International Conference on Universal Personal Communications*, pp. 751-756, Florence, Italy, 5-9 October 1998.

3. Van Nee, R. and R. Prasad, *OFDM for Wireless Multimedia Communications*, Artech House, Boston, 2000.

4. Walrand, J. and P. Varaiya, *High-Performance Communication Networks*, 2nd edition, Morgan Kaufmann, San Fransisco, 2000.

5. Goodman, D. J., *Wireless Personal Communication Systems*, Addison Wesley Longman, Reading, 1997.

6. Lee, S.-J., *Routing and Multicasting Strategies in Wireless Mobile Ad Hoc Networks*, Ph.D. Thesis, University of California, 2000.

7. Royer, E. M., *Routing in Ad Hoc Mobile Networks: On-Demand and Hierarchical Strategies*, Ph.D. Thesis, University of California, 2000.

8. Chakrabarti, S. and A. Mishra, "QoS Issues in Ad Hoc Wireless Networks", *IEEE Communications Magazine*, Vol. 39, No. 2, pp. 142-148, February 2001.

9. Remondo, D. and I. G. Niemegeers, "Ad Hoc Networking in Future Wireless Communications", *Computer Communications*, Vol. 26, No. 1, pp. 36-40, January 2003.

10. Mohapatra, P., J. Li and C. Gui, "QoS in Mobile Ad Hoc Networks", *IEEE Wireless Communications*, Vol. 10, No. 3, pp. 44-52, June 2003.

11. Murthy, C. S. R. and B. S. Manoj, *Ad Hoc Wireless Networks – Architectures and Protocols*, Prentice Hall, Upper Saddle River, 2004.

12. Mohapatra, P., C. Gui and J. Li, "Group Communications in Mobile Ad Hoc Networks", *IEEE Computer*, Vol. 37, No. 2, pp. 52-59, February 2004.

13. Chiang, C.-C., *Wireless Network Multicasting*, Ph.D. Thesis, University of California, 1998.

14. Cordeiro C. M., H. Gossain and D. P. Agrawal, "Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions", *IEEE Network*, Vol. 17, No. 1, pp. 52-59, January 2003.

15. Mihovska, A., J. Pereira and R. Prasad, "Wireless Multimedia: Trends and Requirements", *Proceedings of VTC 2001 Spring – the Semiannual IEEE Vehicular Technology Conference*, pp. 2091-2096, Rhodes, Greece, 6-9 May 2001.

16. Perkins, D. D. and H. D. Hughes, "A Survey on Quality-of-Service Support for Mobile Ad Hoc Networks", *Wireless Communications and Mobile Computing*, Vol. 2, No. 5, pp. 503-513, August 2002.

17. Pattara-Atikom, W. and P. Krishnamurthy, "Distributed Mechanisms for Quality of Service in Wireless LANs", *IEEE Wireless Communications*, Vol. 10, No. 3, pp. 26-33, June 2003.

18. Zhu, H., M. Li, I. Chlamtac and B. Prabhakaran, "A Survey of Quality of Service in IEEE 802.11 Networks", *IEEE Wireless Communications*, Vol. 11, No. 4, pp. 6-14, August 2004.

19. Yang, Y. and R. Kravets, "Contention-Aware Admission Control for Ad Hoc Networks", *IEEE Transactions on Mobile Computing*, Vol. 4, No. 4, pp. 363-377, July 2005.

20. Zhang, Q., W. Zhu, G. J. Wang and Y. Q. Zhang, "Resource Allocation with Adaptive QoS for Multimedia Transmission over W-CDMA Channels", *Proceedings of WCNC 2000 – the IEEE Wireless Communications and Networking Conference*, pp. 179-184, Chicago, USA, 23-28 September 2000.

21. Barry, M., B. J. Hogan and S. McGrath, "Congestion Avoidance in Source Routed Ad Hoc Networks", *Proceedings of the 13th IST Mobile and Wireless Communications Summit*, pp. 682-686, Lyon, France, 27-30 June 2004.

22. Peng, J. and B. Sikdar, "A Multicast Congestion Control Scheme for Mobile Ad-Hoc Networks", *Proceedings of Globecom 2003 – the IEEE Global Communications Conference*, pp. 2860-2864, San Francisco, USA, 1-5 December 2003.

23. Gerla, M. and J. T.-C. Tsai, "Multicluster, Mobile, Multimedia Radio Network", *ACM Wireless Networks*, Vol. 1, No. 3, pp. 255-265, August 1995.

24. Lin, C. R., "On-Demand QoS Routing in Multihop Mobile Networks", *Proceedings of Infocom 2001 – the IEEE Conference on Computer Communications*, pp. 1735-1744, Alaska, USA, 22-26 April 2001.

25. Lin, C. R. and J. S. Liu, "QoS Routing in Ad Hoc Wireless Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1426-1438, August 1999.

26. Zhu, C. and M. S. Corson, "QoS Routing for Mobile Ad Hoc Networks", *Proceedings of Infocom 2002 – the IEEE Conference on Computer Communications*, pp. 958-967, New York, USA, 23-27 June 2002.

27. Chen, Y. S., Y. C. Tseng, J. P. Sheu and P. H. Kuo, "An on-demand, link-state, multi-path QoS routing in a wireless mobile ad-hoc network", *Computer Communications*, Vol. 27, No. 1, pp. 27-40, January 2004.

28. Chen, S. and K. Nahrstedt, "Distributed Quality of Service Routing in Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1488-1505, August 1999.

29. Chen, S., *Routing Support for Providing Guaranteed End-to-End Quality-of-Service*, Ph.D. Thesis, University of Illinois, 1999.

30. Comer, D. E., *Internetworking with TCP/IP – Volume 1: Principles, Protocols, and Architecture*, 3rd edition, Prentice Hall, Upper Saddle River, 1995.

31. Stallings, W., *High-Speed Networks and Internets – Performance and Quality of Service*, 2nd edition, Prentice Hall, Upper Saddle River, 2002.

32. Deering, S. E. and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs", *ACM Transactions on Computer Systems*, Vol. 8, No. 2, pp. 85-110, May 1990.

33. Deering, S. E., "Multicast Routing in Internetworks and Extended LANs", *ACM SIGCOMM Computer Communication Review*, Vol. 18, No. 4, pp. 55-64, August 1988.

34. Ramalho, M., "Intra- and Inter-Domain Multicast Routing Protocols: A Survey and Taxonomy", *IEEE Communications Surveys & Tutorials*, Vol. 3, No. 1, pp. 2-25, First Quarter 2000.

35. Bommaiah, E., M. Liu, A. McAuley and R. Talpade, "AMRoute: Ad Hoc Multicast Routing Protocol", *IETF MANET Working Group Internet Draft (draft-talpade-manet-amroute-00.txt)*, work-in-progress, August 1998.

36. Liu, M., R. R. Talpade, A. McAuley and E. Bommaiah, "AMRoute: Ad Hoc Multicast Routing Protocol", *Center for Satellite and Hybrid Communication Networks Technical Research Report*, University of Maryland, August 1999.

37. Chiang, C.-C., M. Gerla and L. Zhang, "Adaptive Shared Tree Multicast in Mobile Wireless Networks", *Proceedings of Globecom '98 – the IEEE Global Communications Conference*, pp 1817-1822, Sydney, Australia, 8-12 November 1998.

38. Gerla, M., C.-C. Chiang and L. Zhang, "Tree Multicast Strategies in Mobile, Multishop Wireless Networks", *ACM Mobile Networks and Applications*, Vol. 4, No. 3, pp. 193-207, October 1999.

39. Wu, C. W., Y. C. Tay and C. K. Toh. "Ad Hoc Multicast Routing Protocol Utilizing Increasing Id-Numbers (AMRIS) Functional Specification", *IETF MANET Working Group Internet Draft (draft-ietf-manet-amris-spec-00.txt)*, work-in-progress, November 1998.

40. Royer, E. M. and C. E. Perkins, "Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol", *Proceedings of Mobicom '99 – the ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 207-218, Seattle, USA, 15-19 August 1999.

41. Royer, E. M. and C. E. Perkins, "Multicast Ad Hoc On-Demand Distance Vector (MAODV) Routing," *IETF MANET Working Group Internet Draft (draft-ietf-manet-maodv-00.txt)*, work-in-progress, July 2000.

42. Perkins, C. E. and E. M. Royer "Ad-Hoc On-Demand Distance Vector Routing", *Proceedings of WMCSA '99 – the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90-100, New Orleans, USA, 25-26 February 1999.

43. Perkins, C. E., E. M. Royer and S. R. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," *IETF MANET Working Group Internet Draft (draft-ietf-manet-aodv-10.txt)*, work-in-progress, January 2002.

44. Wang, X., F. Li, S. Ishihara and T. Mizuno, " A Multicast Routing Algorithm Based on Mobile Multicast Agents in Ad Hoc Networks", *IEICE Transactions on Communication*, Vol. E84-B, No. 8, pp. 2087-2094, August 2001.

45. Shen, C.-C. and C. Jaikaeo, "Ad Hoc Multicast Routing Algorithm with Swarm Intelligence", *ACM Mobile Networks and Applications*, Vol. 10, No. 1-2, pp. 47-59, February 2005.

46. Sinha, P., R. Sivakumar and V. Bharghavan, "MCEDAR: Multicast Core-Extraction Distributed Ad Hoc Routing", *Proceedings of WCNC '99 – the IEEE Wireless Communications and Networking Conference*, pp. 1313-1317, New Orleans, USA, 21-24 September 1999.

47. Sinha, P., R. Sivakumar and V. Bharghavan, "CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm", *Proceedings of Infocom '99 – the IEEE Conference on Computer Communications*, pp. 202-209, New York, USA, 23-25 March 1999.

48. Sivakumar, R., P. Sinha and V. Bharghavan, "CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm", *IEEE Journal of Selected Areas in Communications*, Vol. 17, No.8, pp. 1454-1465, August 1999.

49. Ozaki, T., J. B. Kim and T. Suda, "Bandwidth-Efficient Multicast Routing Protocol for Ad Hoc Networks", *Proceedings of ICCCN '99 – the IEEE International Conference on Computer Communications and Networks*, pp. 10-17, Boston, USA, 11-13 October 1999.

50. Ozaki, T., J. B. Kim and T. Suda, "Bandwidth-Efficient Multicast Routing for Multihop, Ad Hoc Wireless Networks", *Proceedings of Infocom 2001– the IEEE*

*Conference on Computer Communications*, pp. 1182-1191, Alaska, USA, 22-26 April 2001.

51. Ji, L. and M. S. Corson, "Differential Destination Multicast (DDM) Specification", *IETF MANET Working Group Internet Draft (draft-ietf-manet-ddm-00.txt)*, work-in-progress, July 2000.

52. Ji, L. and M. S. Corson, "Differential Destination Multicast – A MANET Multicast Routing Protocol for Small Groups", *Proceedings of Infocom 2001 – the IEEE Conference on Computer Communications*, pp. 1192-1201, Alaska, USA, 22-26 April 2001.

53. Ji, L. and M. S. Corson, "Explicit Multicasting for Mobile Ad Hoc Networks", *ACM Mobile Networks and Applications*, Vol. 8, No. 5, pp. 535-549, October 2003.

54. Toh, C. K., G. Guichala and S. Bunchua, "ABAM: On-Demand Associativity-Based Multicast Routing for Ad Hoc Mobile Networks", *Proceedings of VTC 2000 Fall – the Semiannual IEEE Vehicular Technology Conference*, pp. 987-993, Boston, USA, 24-28 September 2000.

55. Toh, C. K., *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Prentice Hall, Upper Saddle River, 2002.

56. Toh, C. K., "Associativity-Based Routing for Ad Hoc Mobile Networks", *Wireless Communications Journal*, Vol. 4, No. 2, pp. 1-36, March 1997.

57. Devarapalli, V., A. A. Selçuk and D. Sidhu, "MZR: A Multicast Protocol for Mobile Ad Hoc Networks", *IETF MANET Working Group Internet Draft (draft-vijay-manet-mzr-01.txt)*, work-in-progress, July 2001.

58. Haas, Z. J., M. R. Pearlman and P. Samar, "Zone Routing Protocol (ZRP)" *IETF MANET Working Group Internet Draft (draft-ietf-manet-zrp-04.txt)*, work-in-progress, January 2001.

59. Das, S. K., B. S. Manoj and C. S. R. Murthy, "Weight-Based Multicast Routing Protocol for Ad Hoc Wireless Networks", *Proceedings of Globecom 2002 – the IEEE Global Communications Conference*, pp. 17-21, Taipei, Taiwan, 17-21 November 2002.

60. Sisodia, R. S., I. Karthigeyan, B. S. Manoj and C. S. R. Murthy, "A Preferred Link-Based Multicast Protocol for Wireless Mobile Ad Hoc Networks", *Proceedings of ICC 2003 – the IEEE International Conference on Communications*, pp. 2213-2217, Anchorage, USA, 11-15 May 2003.

61. Sisodia, R. S., B. S. Manoj and C. S. R. Murthy, "A Preferred Link-Based Routing Protocol for Ad Hoc Wireless Networks", *Journal of Communications and Networks*, Vol. 4, No. 1, pp. 14-21, March 2002.

62. Chiang, C.-C., M. Gerla and L. Zhang, "Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks", *ACM/Baltzer Journal of Cluster Computing*, Vol. 1, No. 2, pp. 187-196, November 1998.

63. Madruga, E. L. and J. J. Garcia-Luna-Aceves, "Multicasting Along Meshes in Ad-Hoc Networks", *Proceedings of ICC '99 – the IEEE International Conference on Communications*, pp. 314-318, Vancouver, Canada, 6-10 June 1999.

64. Garcia-Luna-Aceves, J. J. and E. L. Madruga, "A Multicast Routing Protocol for Ad Hoc Networks", *Proceedings of Infocom '99 – the IEEE Conference on Computer Communications*, pp 784-792, New York, USA, 23-25 March 1999.

65. Garcia-Luna-Aceves, J. J. and E. L. Madruga, "The Core-Assisted Mesh Protocol", *IEEE Journal of Selected Areas in Communications*, Vol. 17, No.8, pp. 1380-1394, August 1999.

66. Madruga, E. L. and J. J. Garcia-Luna-Aceves, "Scalable Multicasting: The Core-Assisted Mesh Protocol", *ACM Mobile Networks and Applications*, Vol. 6, No. 2, pp. 151-165, March 2001.

67. Lee, S. J., M. Gerla and C. C. Chiang, "On-Demand Multicast Routing Protocol", *Proceedings of WCNC '99 – the IEEE Wireless Communications and Networking Conference*, pp. 1298-1302, New Orleans, USA, 21-24 September 1999.

68. Bae, H. B., S. J. Lee, W. Su and M. Gerla, "The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocol in Multihop Wireless Networks", *IEEE Network*, Vol. 14, No. 1, pp. 70-77, January 2000.

69. Lee, S. J., W. Su, J. Hsu and M. Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks", *IETF MANET Working Group Internet Draft (draft-ietf-manet-odmrp-02.txt)*, work-in-progress, January 2000.

70. Lee, S. J., W. Su and M. Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks", *ACM Mobile Networks and Applications*, Vol. 7, No. 6, pp. 441-453, December 2002.

71. Tang, K. and M. Gerla, "Reliable Multicast of the On-Demand Multicast Routing Protocol", *Proceedings of SCI 2001 – the International Conference on Systemics, Cybernetics and Informatics*, Orlando, USA, 22-25 July 2001.

72. Cai, S., L. Wang and X. Yang, "An ad hoc multicast protocol based on passive data acknowledgement", *Computer Communications*, Vol. 27, No. 18, pp. 1812-1824, December 2004.

73. Lee, S. and C. Kim, "Neighbor Supporting Ad Hoc Multicast Routing Protocol", *Proceedings of MobiHOC 2000 – the IEEE/ACM Workshop on Mobile Ad Hoc Networking & Computing*, pp. 37-50, Boston, USA, 11 August 2000.

74. Lee, S. and C. Kim, "A new wireless ad hoc multicast routing protocol", *Computer Networks*, Vol. 38, No. 2, pp. 121-135, February 2002.

75. Das, S. K., B. S. Manoj and C. S. R. Murthy, "A Dynamic Core-Based Multicast Routing Protocol for Ad Hoc Wireless Networks", *Proceedings of MobiHOC 2002 –*

*the ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 24-35, Lausanne, Switzerland, 9-11 June 2002.

76. Luo, J., P. Th. Eugster and J.-P. Hubaux, "Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks", *Proceedings of Infocom 2003 – the IEEE Conference on Computer Communications*, pp. 2229-2239, San Fransisco, USA, 1-3 April 2003.

77. Luo, J., P. Th. Eugster and J.-P. Hubaux, "Probabilistic reliable multicast in ad hoc networks", *Ad Hoc Networks*, Vol. 2, No. 4, pp. 369-386, October 2004.

78. Vaishampayan, R. and J. J. Garcia-Luna-Aceves, "Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks", *Proceedings of MASS 2004 – the IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pp. 304-313, Fort Lauderdale, USA, 24-27 October 2004.

79. Sajama, S. and Z. J. Haas, "Independent-Tree Ad Hoc Multicast Routing (ITAMAR)", *Proceedings of VTC 2001 Fall – the Semiannual IEEE Vehicular Technology Conference*, pp. 600-604, Atlantic City, USA, 7-11 October 2001.

80. Sajama, S. and Z. J. Haas, "Independent-Tree Ad Hoc Multicast Routing (ITAMAR)", *ACM Mobile Networks and Applications*, Vol. 8, No. 5, pp. 551-566, October 2003.

81. Chen, Y. S., Y. W. Ko and T. L. Lin, "A Lantern-Tree-Based QoS Multicast Protocol for Wireless Ad Hoc Networks", *Proceedings of ICCCN 2002 – the IEEE International Conference on Computer Communications and Networks*, pp. 242-247, Miami, USA, 14-16 October 2002.

82. Chen, Y. S. and Y. W. Ko, "A Lantern-Tree-Based QoS On-Demand Multicast Protocol for a Wireless Mobile Ad Hoc Network", *IEICE Transactions on Communications*, Vol. E87-B, No. 3, pp. 717-726, March 2004.

83. Pompili, D. and M. Vittucci, “A Probabilistic Predictive Multicast Algorithm in Ad Hoc Networks (PPMA)”, *Proceedings of Med-Hoc-Net 2004 – the Mediterranean Ad Hoc Networking Workshop*, Muğla, Turkey, 27-30 June 2004.

84. Pompili, D. and M. Vittucci, "PPMA, a probabilistic predictive multicast algorithm for ad hoc networks", *Ad Hoc Networks*, article-in-press, October 2005.

85. IEEE Communications Society LAN/MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-1999 (ISO/IEC 8802-11: 1999), 1999.

86. Cain, B., S. Deering, I. Kouvelas, B. Fenner and A. Thyagarajan, *Internet Group Management Protocol, Version 3*, RFC 3376, October 2002.

87. Deering, S., W. Fenner and B. Habermann, *Multicast Listener Discovery (MLD) for IPv6*, RFC 2710, October 1999.

88. Maltz, D. A. and J. Broch, “Lessons from a Full-Scale Multihop Wireless Ad Hoc Network Testbed”, *IEEE Personal Communications*, Vol. 8, No. 1, pp. 8-15, February 2001.

89. Bae, S. H., S.-J. Lee and M. Gerla, “Multicast Protocol Implementation and Validation in an Ad hoc Network Testbed”, *Proceedings of ICC 2001 – the IEEE International Conference on Communications*, pp. 3196-3200, Helsinki, Finland, 11-14 June 2001.

90. Jelger, C. and T. Noel, “Multicast for Mobile Hosts in IP Networks: Progress and Challenges”, *IEEE Wireless Communications*, Vol. 9, No. 5, pp. 58-64, October 2002.

91. Ko, B., S. Ahn, N. Kim, Y. Lee and D. Lee, “An Integration Scheme of Overlay Multicast Protocol for Wired Networks and Wireless Ad Hoc Networks”, *Proceedings of ICWN 2005 – the International Conference on Wireless Networks*, pp. 57-62, Las Vegas, USA, 27-30 June 2005.

92. Chen, K., Y. Xue, S. H. Shah and K. Nahrstedt, "Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks", *Computer Communications*, Vol. 27, No. 10, pp. 923-934, June 2004.

93. Nahrstedt, K., S. H. Shah and K. Chen, "Cross-Layer Architectures for Bandwidth Management in Wireless Networks", in M. Cardei, I. Cardei and D.Z. Du (eds.), *Resource Management in Wireless Networking*, pp. 41-62, Springer Verlag, Berlin, 2005.

94. Shah, S. H., K. Chen and K. Nahrstedt, "Available Bandwidth Estimation in IEEE 802.11-based Wireless Networks", *Proceedings of BEst 2003 – the ISMA/CAIDA Workshop on Bandwidth Estimation*, San Diego, USA, 9-10 December 2003.

95. Yang, Y., J. Wang and R. Kravets, "Achievable Bandwidth Prediction in Multihop Wireless Networks", *Department of Computer Science Technical Report*, University of Illinois at Urbana-Champaign, December 2003.

96. Shah, S. H., K. Chen and K. Nahrstedt, "Dynamic Bandwidth Management for Single-Hop Ad Hoc Wireless Networks", *Proceedings of PerCom 2003 – the IEEE International Conference on Pervasive Computing and Communications*, pp. 195-203, Dallas, USA, 23-26 March 2003.

97. Shah, S. H., K. Chen and K. Nahrstedt, "Dynamic Bandwidth Management in Single-Hop Ad Hoc Wireless Networks", *ACM Mobile Networks and Applications*, Vol. 10, No. 1, pp. 199-217, February 2005.

98. Xue, Q. and A. Ganz, "Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks", *Journal of Parallel and Distributed Computing*, Vol. 63, No. 2, pp. 154-165, February 2003.

99. Corson, S. and J. Macker, *Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, RFC 2501, January 1999.

100. OPNET Technologies Inc., Bethesda, MD, USA, http://www.opnet.com.

101. Camp, T., J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research", *Wireless Communications and Mobile Computing*, Vol. 2, No. 5, pp. 483-502, August 2002.

102. Bettstetter, C., G. Resta and P. Santi, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks", *IEEE Transactions on Mobile Computing*, Vol. 2, No. 3, pp 257-269, July 2003.

103. Sollins, K., *The TFTP Protocol (Revision 2)*, RFC 1350, July 1992.

104. Kleinrock, L. and J. Silvester, "Optimum Transmission Radii for Packet Radio Networks or Why Six is A Magical Number", *Proceedings of the IEEE National Telecommunication Conference*, pp. 4.3.1-4.3.5, Birmingham, USA, December 1978.

105. Takagi, H. and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals", *IEEE Transactions on Communications*, Vol. COM-32, No. 3, pp. 246-257, March 1984.

106. Bür, K. and C. Ersoy, "Ad Hoc Quality of Service Multicast Routing with Objection Queries for Admission Control", *European Transactions on Telecommunications*, article-in-press, September2005.

107. Bür, K. and C. Ersoy, "Ad Hoc Quality of Service Multicast Routing", *Computer Communications*, Vol. 29, No. 1, pp. 136-148, December 2005.

108. Bharghavan, V., S. Lu and T. Nandagopal, "Fair Queuing in Wireless Networks: Issues and Approaches", *IEEE Personal Communications*, Vol. 6, No. 1, pp. 44-53, February 1999.

109. Martinez, I. and J. Altuna, "A Cross-Layer Design for Ad Hoc Wireless Networks with Smart Antennas and QoS Support", *Proceedings of PIMRC 2004 – the IEEE*

*International Symposium on Personal, Indoor and Mobile Radio Communications*, Barcelona, Spain, 5-8 September 2004.

110. Braden, R., D. Clark and S. Shenker, *Integrated Services in the Internet Architecture: An Overview*, RFC 1633, June 1994.

111. Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang and W.Weiss, *An Architecture for Differentiated Services*, RFC 2475, December 1998.

112. Braden, R., L. Zhang, S. Berson, S. Herzog and S. Jamin, *Resource reSerVation Protocol (RSVP) -- Version 1 Functional Specification*, RFC 2205, September 1997.

113. Lee, S.-B., G.-S. Ahn, X. Zhang and A. T. Campbell, "INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks", *Journal of Parallel and Distributed Computing*, Vol. 60, No. 4, pp. 374-406, April 2000.

114. Ahn, G.-S., A. T. Campbell, A. Veres and L.-H. Sun, "Supporting Service Differentiation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)", *IEEE Transactıons on Mobile Computing*, Vol. 1, No. 3, July 2002.

115. Yang, Y. and R. Kravets, " Distributed QoS Guarantees for Realtime Traffic in Ad Hoc Networks", *Proceedings of SECON 2004 – the IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, pp. 118-127, Santa Clara, USA, 4-7 October 2004.

116. Fattah, H. and C. Leung, "An Overview of Scheduling Algorithms in Wireless Multimedia Networks", *IEEE Wireless Communications*, Vol. 9, No. 5, pp. 76-83, October 2002.

117. Bennett, J. C. R. and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 675-689, October 1997.

118. Chen, S. and K. Nahrstedt, “Hierarchical Scheduling for Multiple Classes of Applications in Connection-Oriented Integrated-Service Networks”, *Proceedings of ICMCS ’99 – the IEEE International Conference on Multimedia Computing and Systems*, pp. 153-158, Florence, Italy, 7-11 June 1999.