Decentralized Key Management for Large Dynamic Multicast Groups using Distributed Balanced Trees

Thesis by

Junaid Haroon MSCS018

Supervised by

Mr Shafiq ur Rahman

Flow of Presentation

- Background
- Proposed Scheme
- Analysis
- References

Background

- Two Party Secure Communication
- Multicast Groups (Without Security)
- Secure Multicast
- Key Management for Secure Multicast
- Logical Key Hierarchy
- Problems in Existing Scheme

Two Party Secure Communication

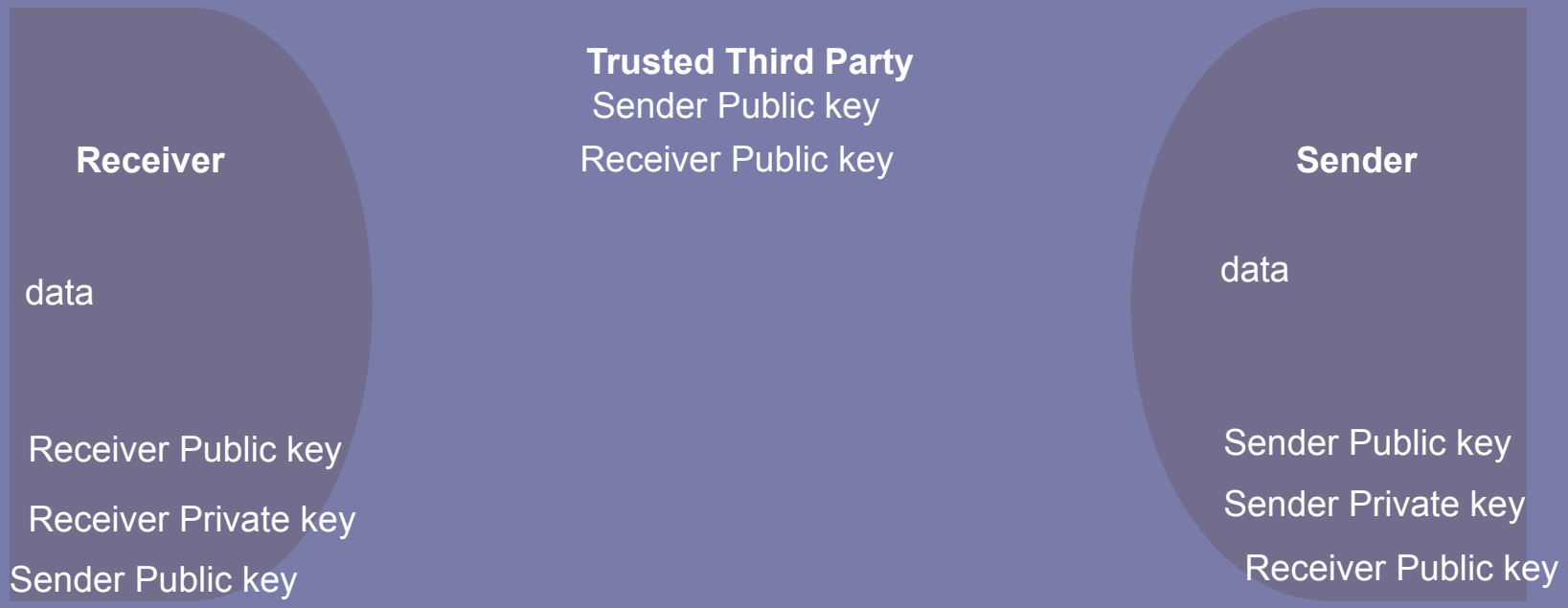
- Both parties generate a pair of keys such that data encrypted with one of them can be decrypted with the other

Two Party Secure Communication

- Both parties register one key with a trusted third party. This key is called the public key while the other is called private key.

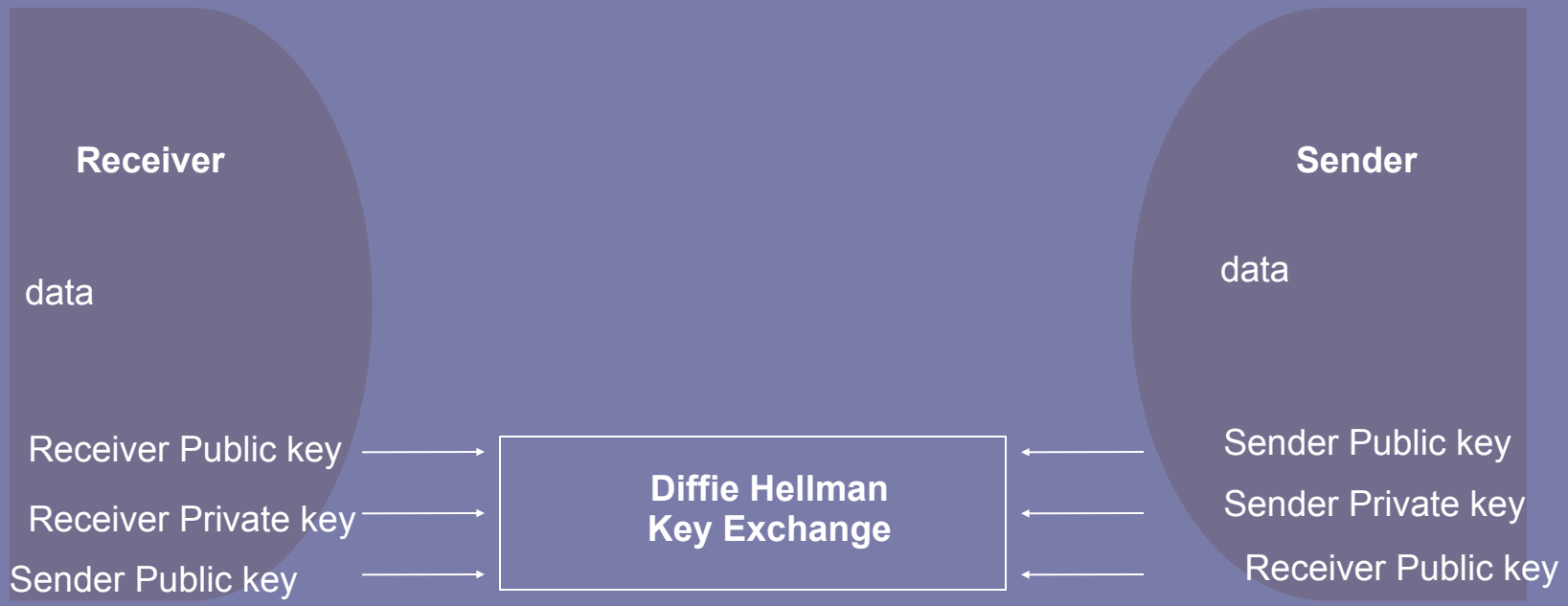
Two Party Secure Communication

- Both parties acquire the public key of the other party from the trusted third party



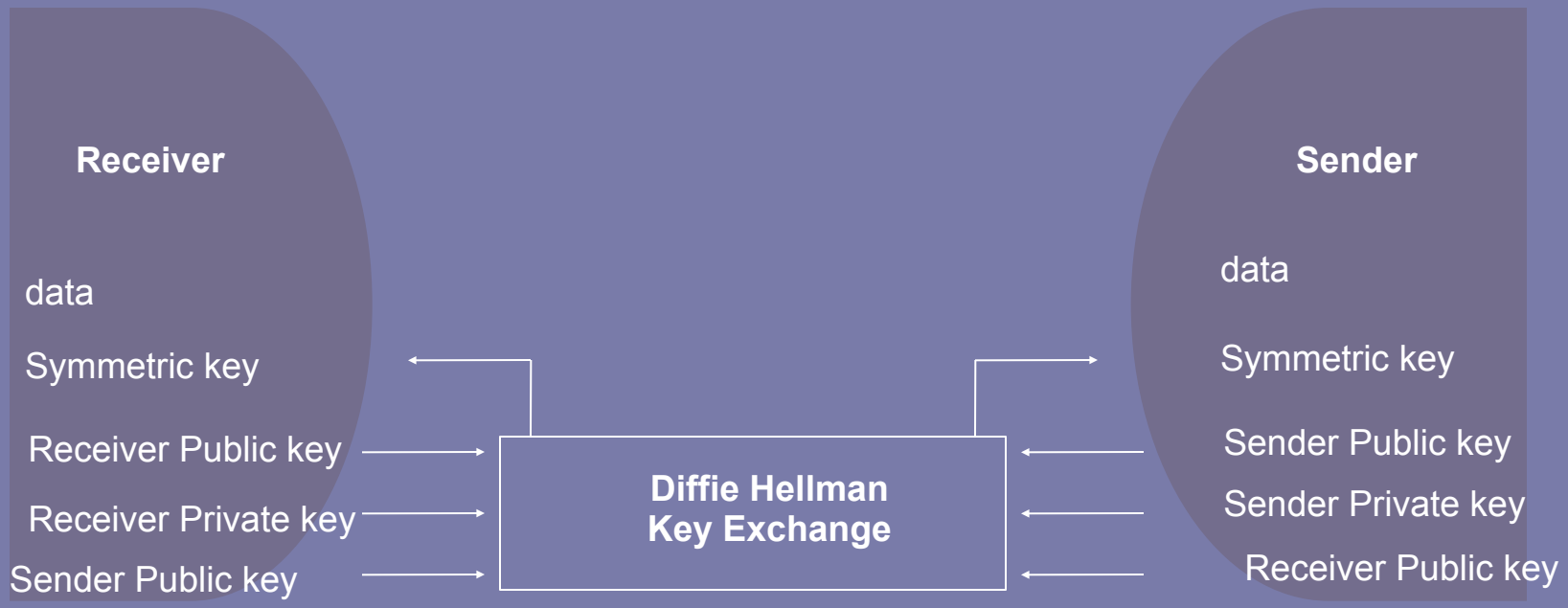
Two Party Secure Communication

- Both parties use the Deffie Hellman protocol to arrive at a common key at both ends



Two Party Secure Communication

- Both parties use the Deffie Hellman protocol to arrive at a common key at both ends

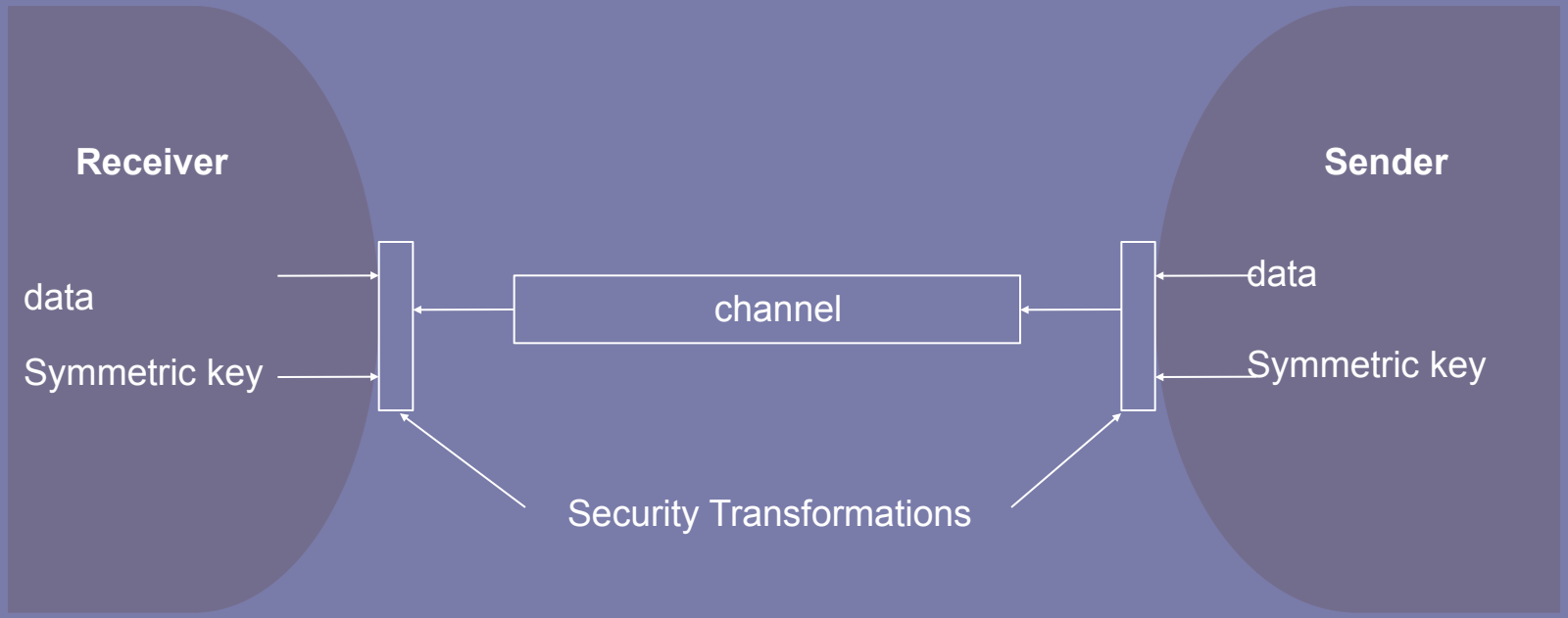


Two Party Secure Communication

- Both parties use the Deffie Hellman protocol to arrive at a common key at both ends

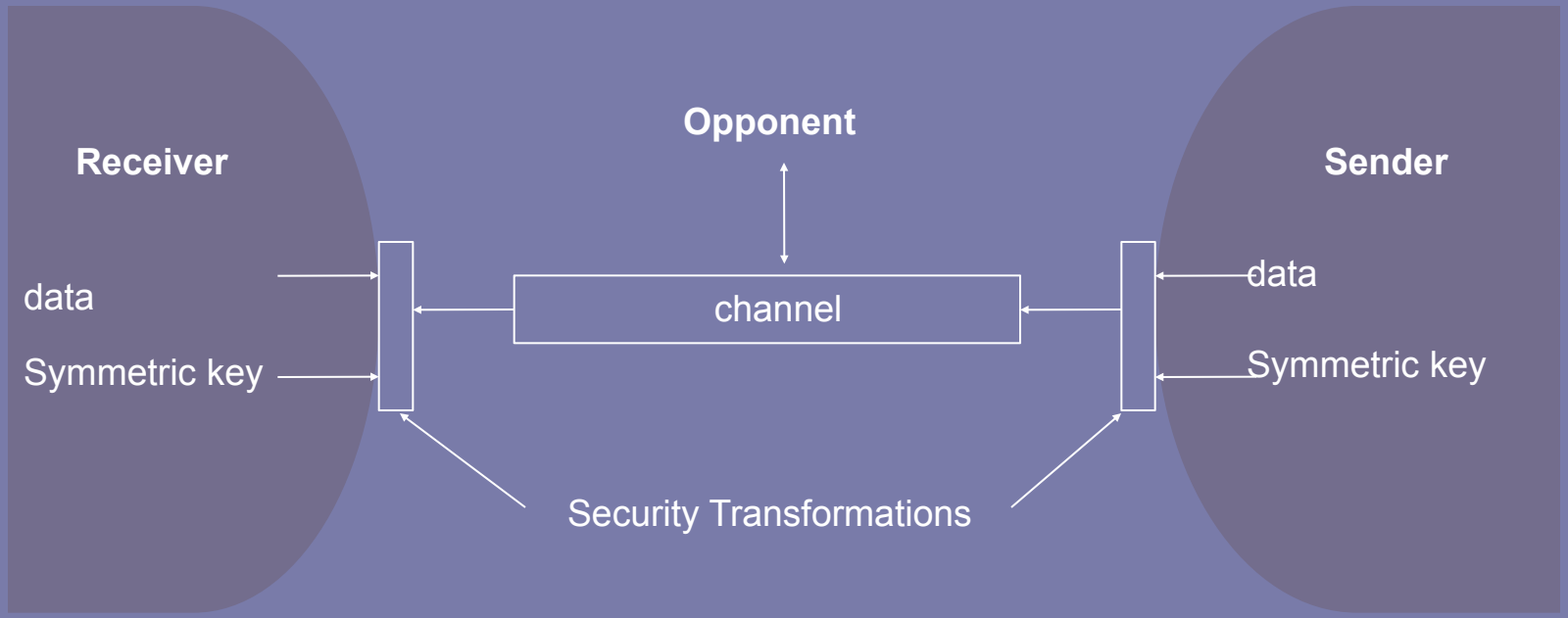
Two Party Secure Communication

- Both parties use symmetric encryption using this generated key to transfer data over the public channel

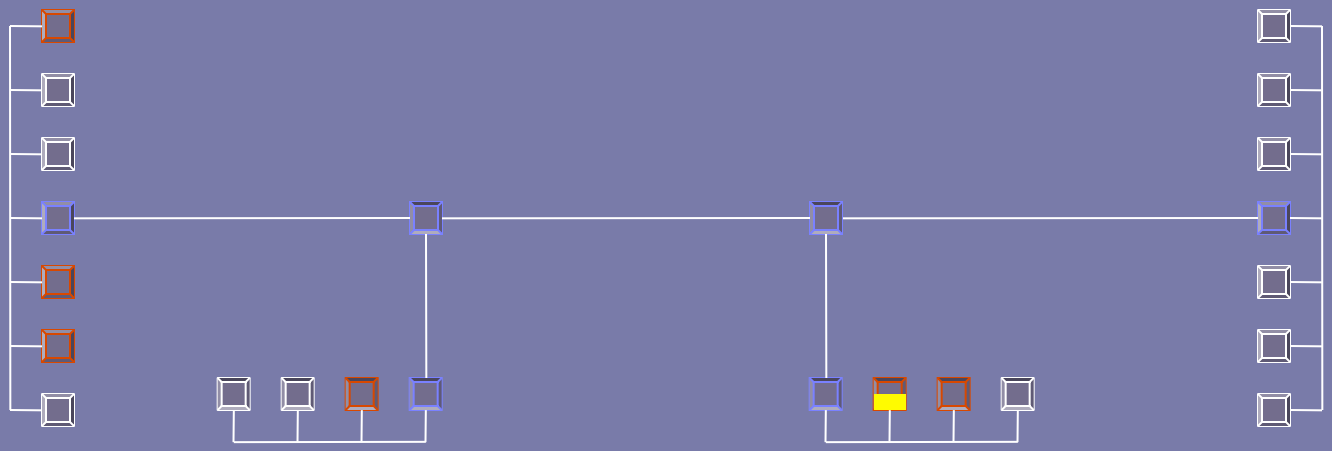


Two Party Secure Communication

- The opponent is unable to decipher encrypted data passing over the channel

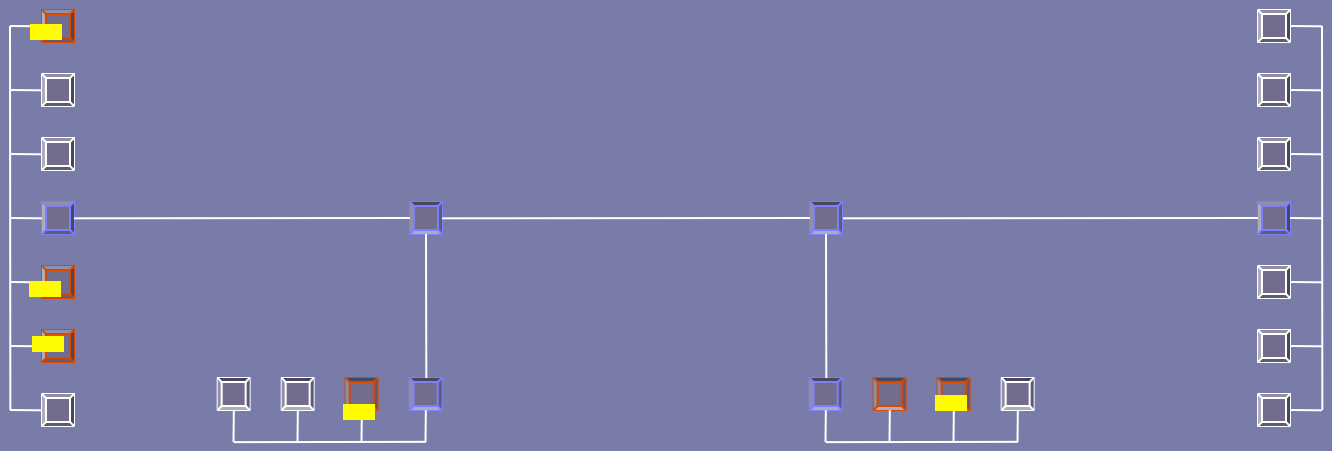


Multicast Groups (Without Security)



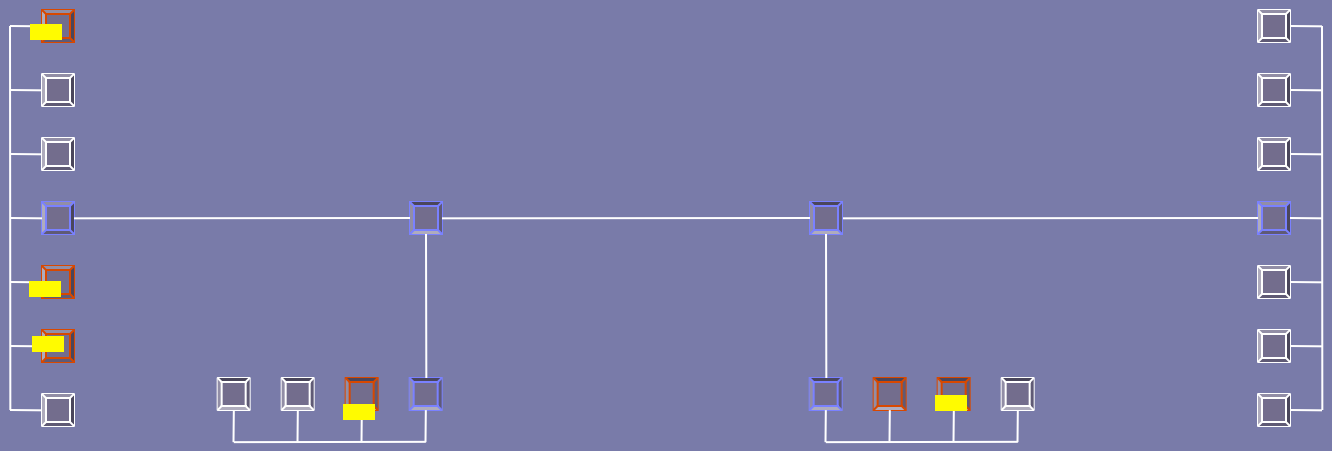
Multicast Groups (Without Security)

- Multicast is the delivery of a packet to all hosts which have shown interest in receiving it



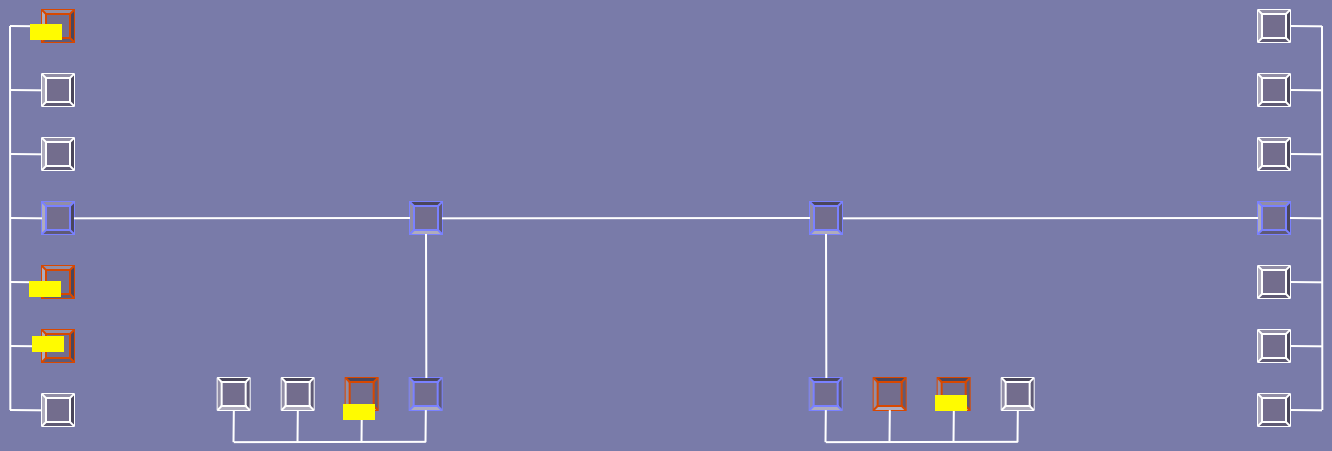
Multicast Groups (Without Security)

- Multicast capable routers construct each group's delivery tree
- Hosts can join or leave a multicast group by informing their nearest router



Multicast Groups (Without Security)

- In multicast there is no authentication or authorization for group membership and for sending data to a group

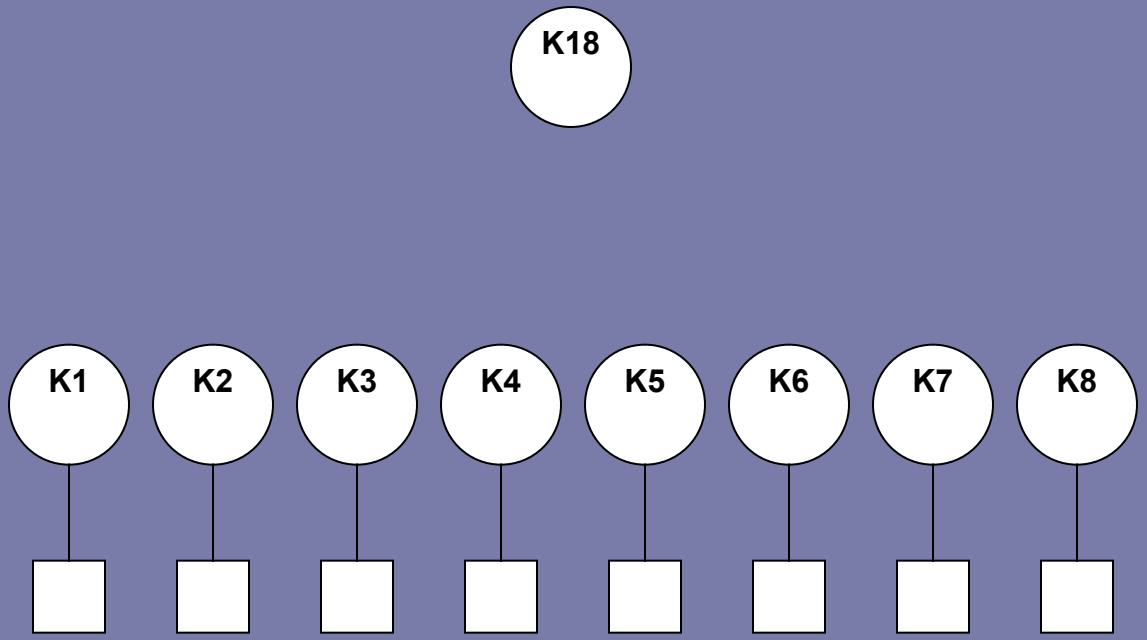


Secure Multicast

- All transmitted data must be encrypted and only group members should possess the key used to encrypt and decrypt, also called the group key
- After every member join or leave in the secure group, key must be changed otherwise
  - A joining member can decrypt accumulated traffic
  - A leaving member can continue to decrypt data
- The mechanism by which the new key is generated and distributed is called key management
- The message containing the new key is called key update

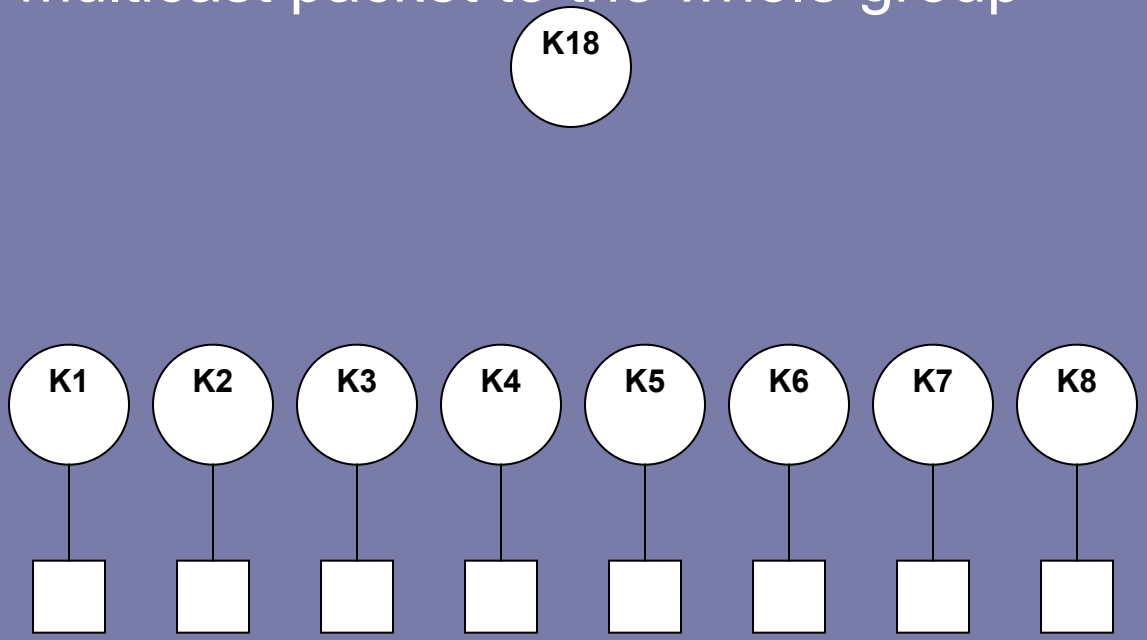
Key Management for Multicast Groups

- In simple schemes key updates are sent in a single multicast packet containing the group key encrypted by each member's public key. Size of key update is proportional to group size



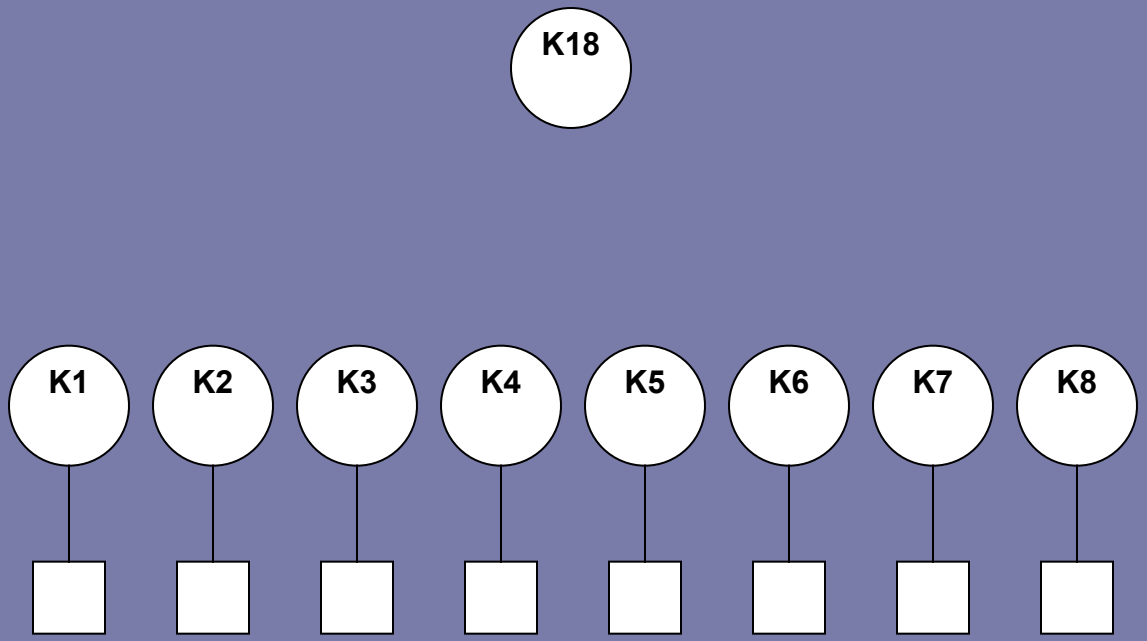
Key Management for Multicast Groups

- For example, if M8 wants to leave the group, the new K18 is sent encrypted with each of K1 through K7 but not with K8
- These seven encrypted copies of K18 are sent in a single multicast packet to the whole group



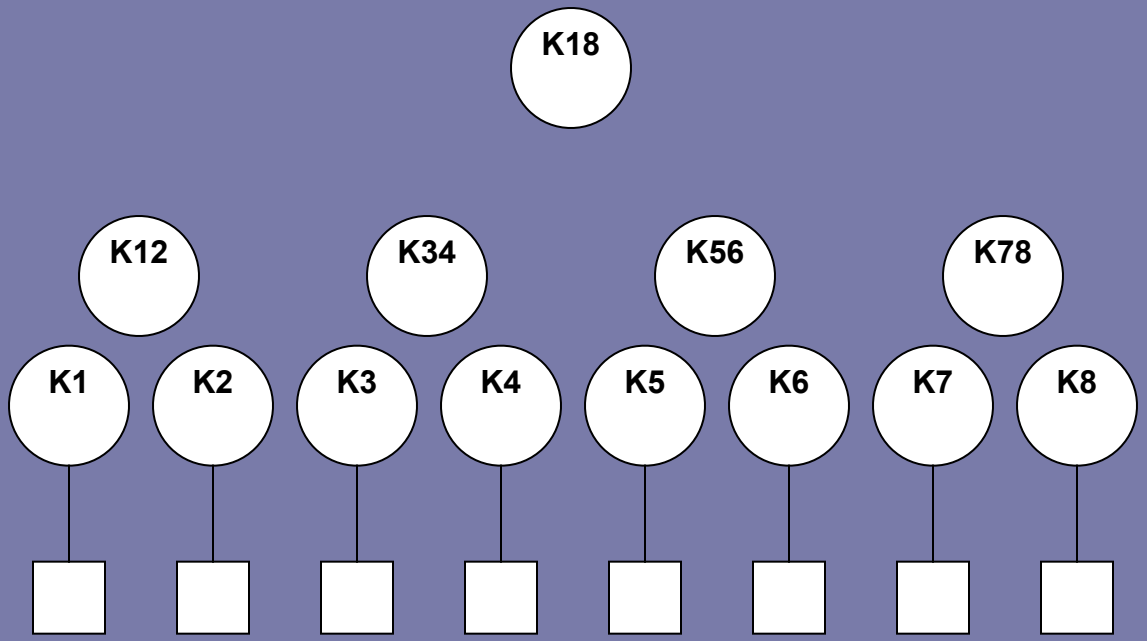
Key Management for Multicast Groups

- If every pair of hosts know a common key, the required encryptions reduce



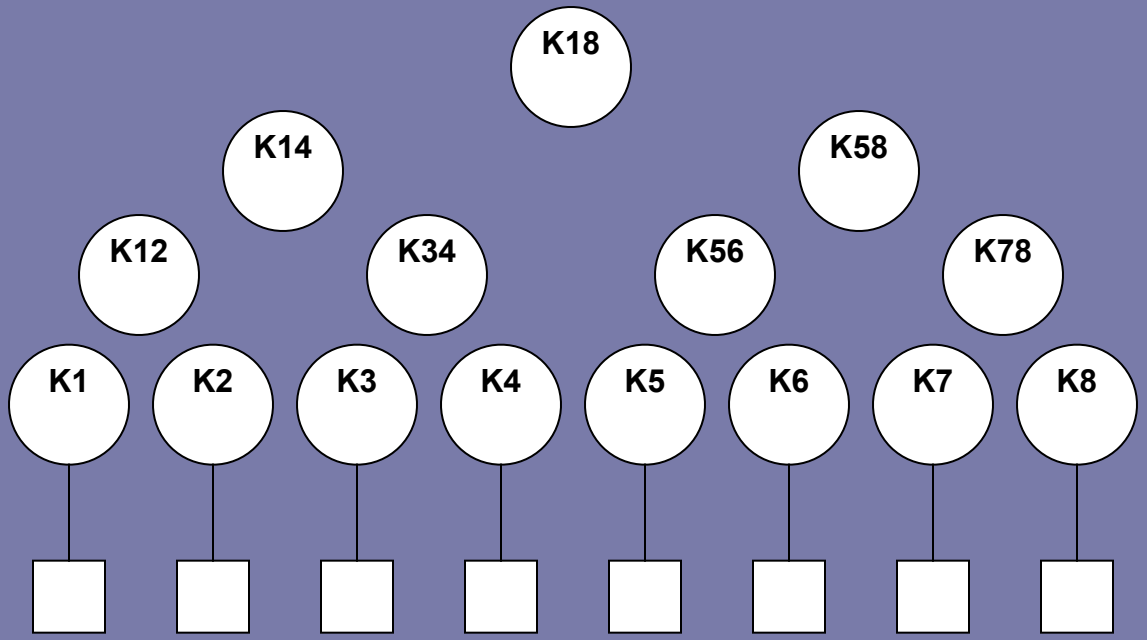
Key Management for Multicast Groups

- If every pair of hosts know a common key, the required encryptions reduce



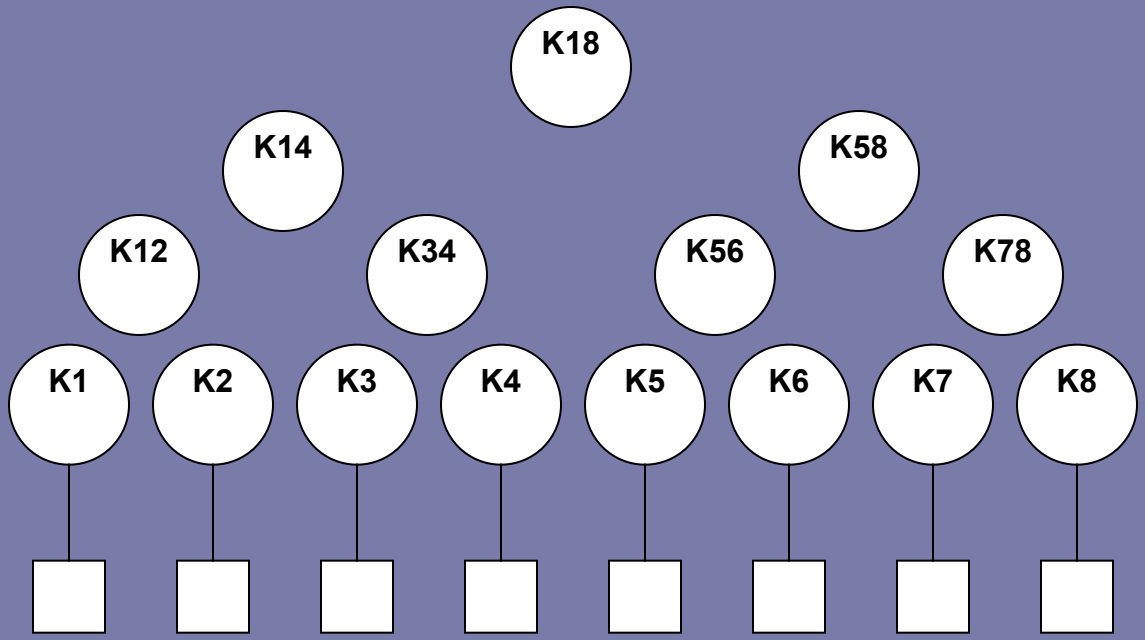
Key Management for Multicast Groups

- If every pair of hosts know a common key, the required encryptions reduce
- If this is extended to form a tree the number of encryptions become logarithmic



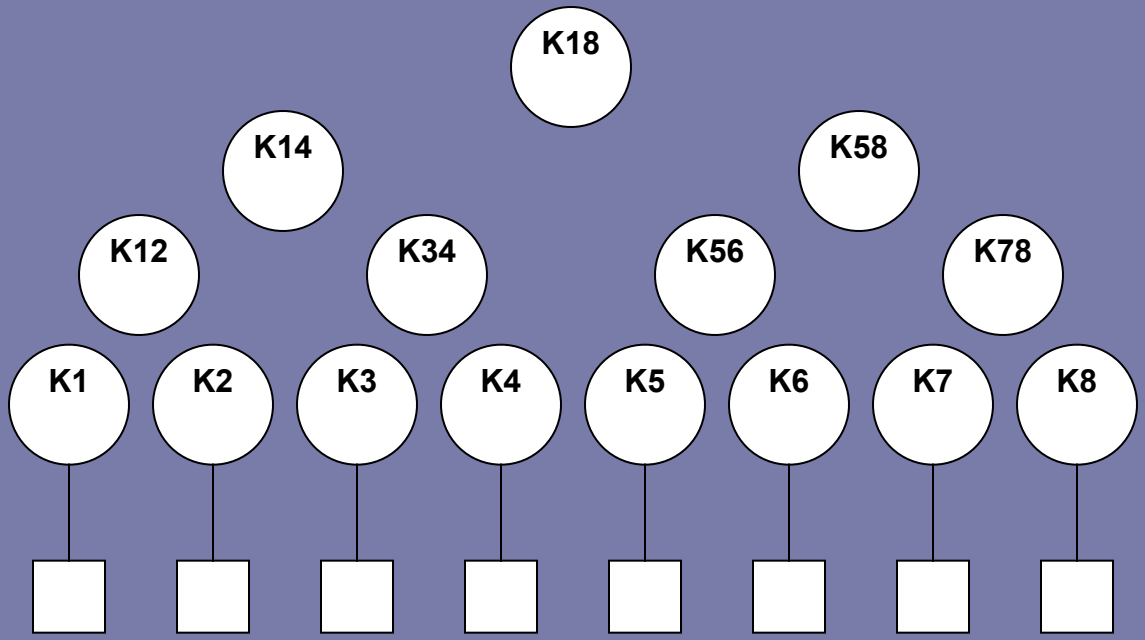
Key Management for Multicast Groups

- However two more keys need to be changed which were known by the leaving member
- The whole process goes like this



Key Management for Multicast Groups

- A total of five encryptions were needed
- Size of message is 2 lg n
- The scheme is called Logical Key Hierarchy



Logical Key Hierarchy (LKH)

- Each member knows all keys on its path to the root
- On member join or leave all keys on that member's path to the root are updated
- A changed key is sent encrypted by both its children keys therefore message size becomes 2 log n

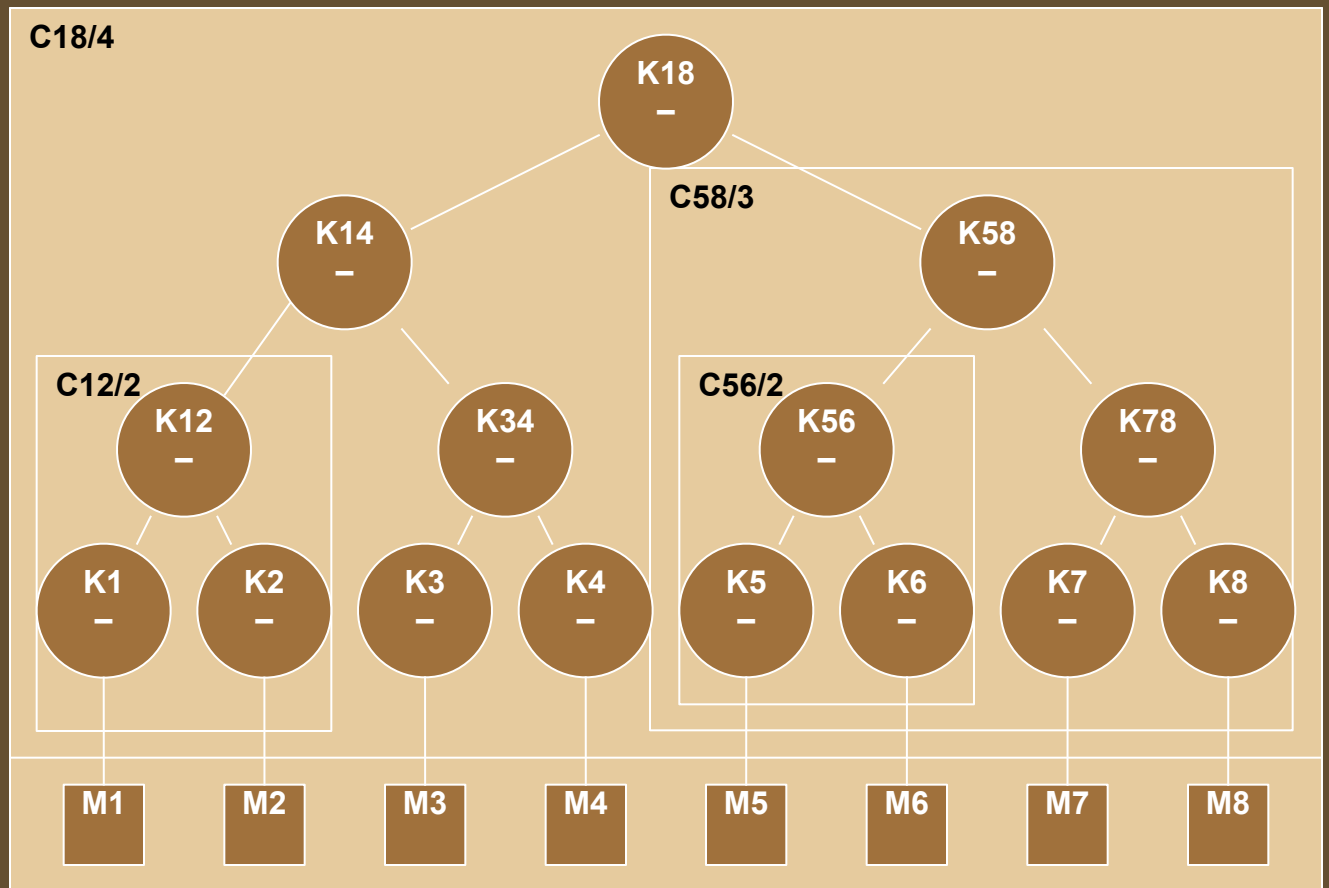
Problems in Existing Scheme

- Single controller responsible for storing keys for the whole group and generating key updates for them
  - Performance deteriorates as group size increases
- Key update message size is logarithmic only when the key tree is balanced which is not guaranteed in this scheme
  - Performance deteriorates as tree gets out of balance

Proposed Scheme

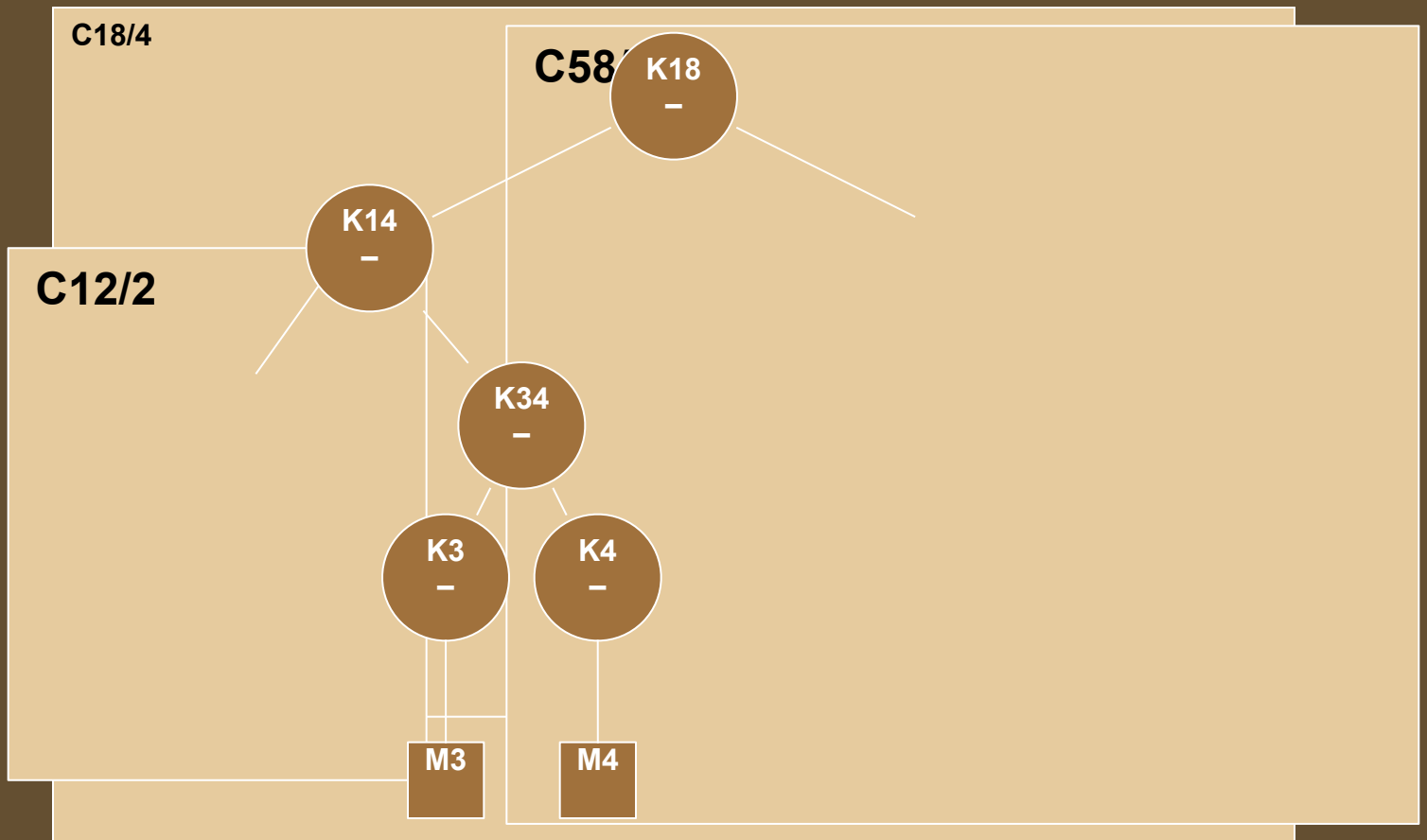
- Distributed Nature of Key Tree
- Balancing of Key Tree
- Key Updates
- Member Joins
- Member Leaves

Distributed Nature of Key Tree



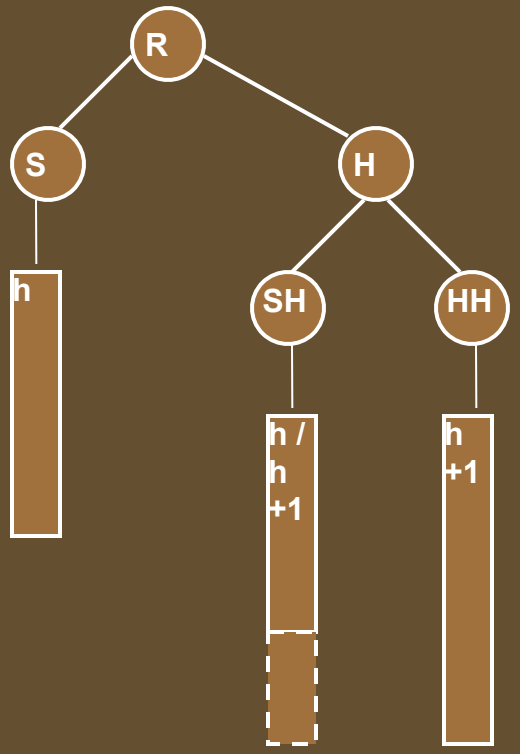
Distributed Nature of Key Tree

- Every controller has a key as allotted to it by the parent and another key that it generates to be used in the key tree



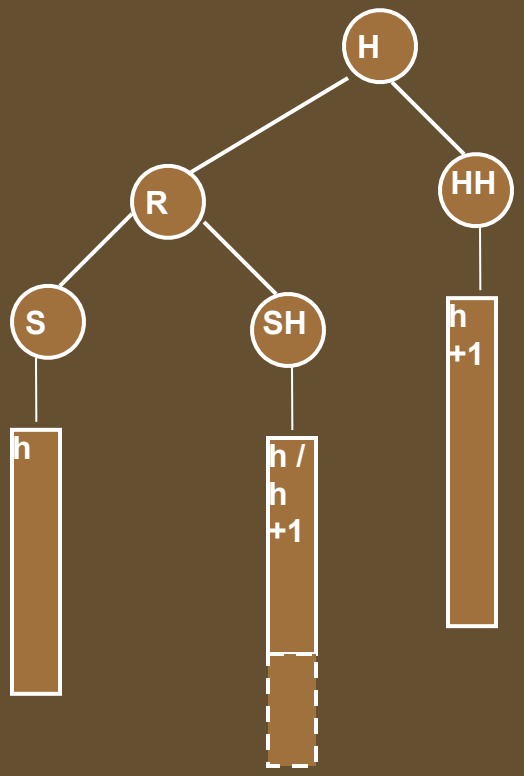
Balancing of Key Tree

- There is no order in members so there is a single case of rotation which is like single rotation in an AVL tree



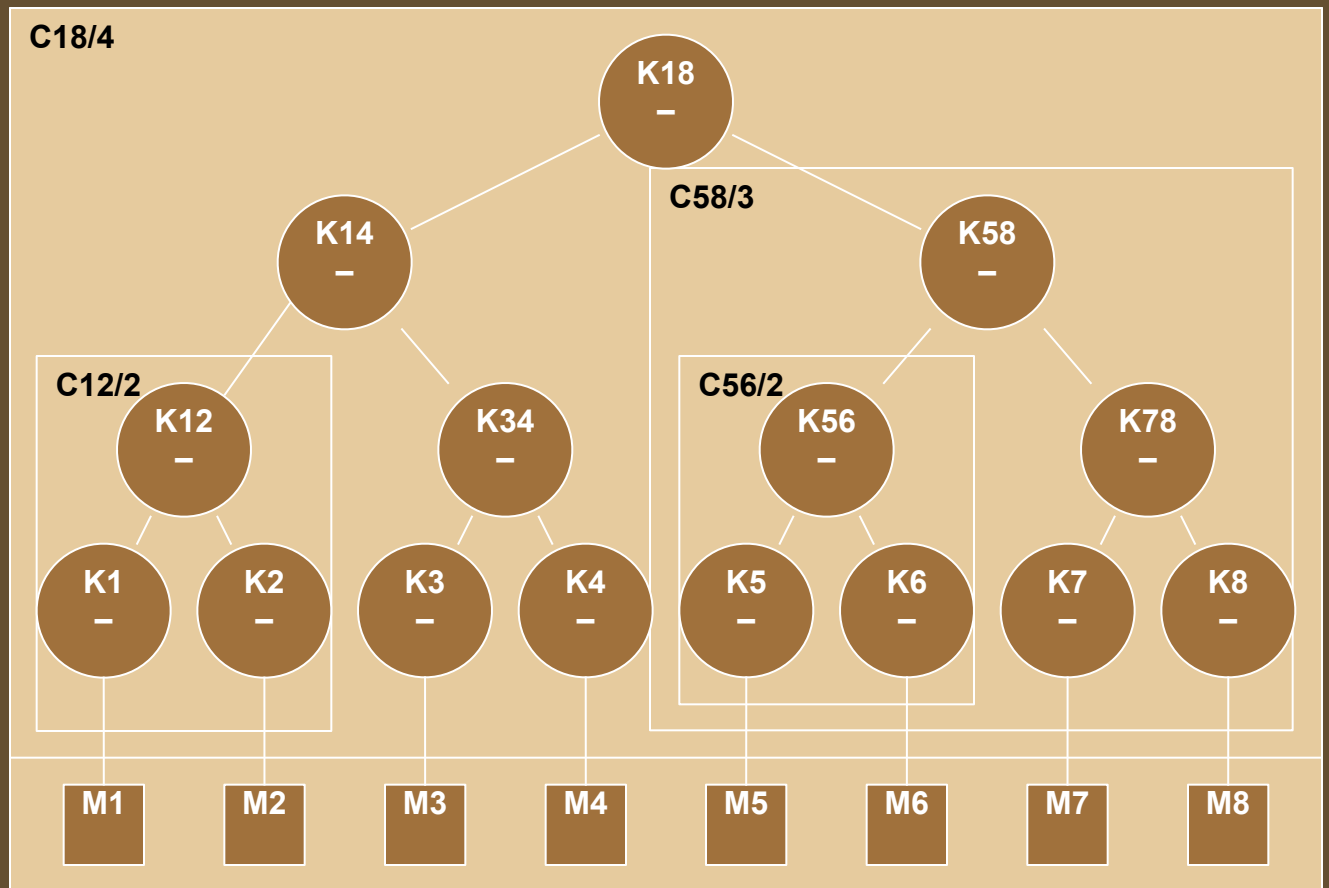
Balancing of Key Tree

- There is no order in members so there is a single case of rotation which is like single rotation in an AVL tree

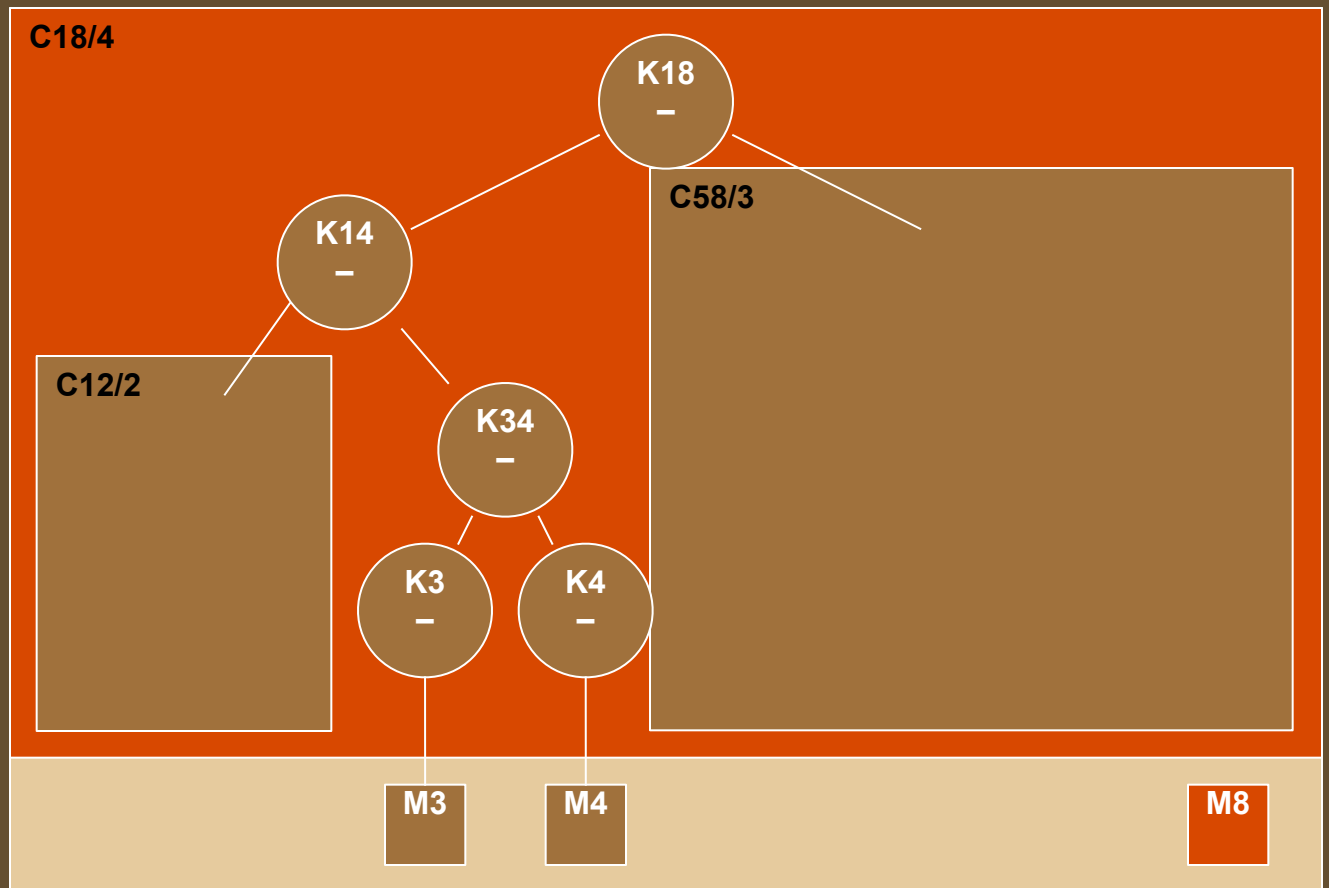


Key Updates

- Assume all keys need to updated

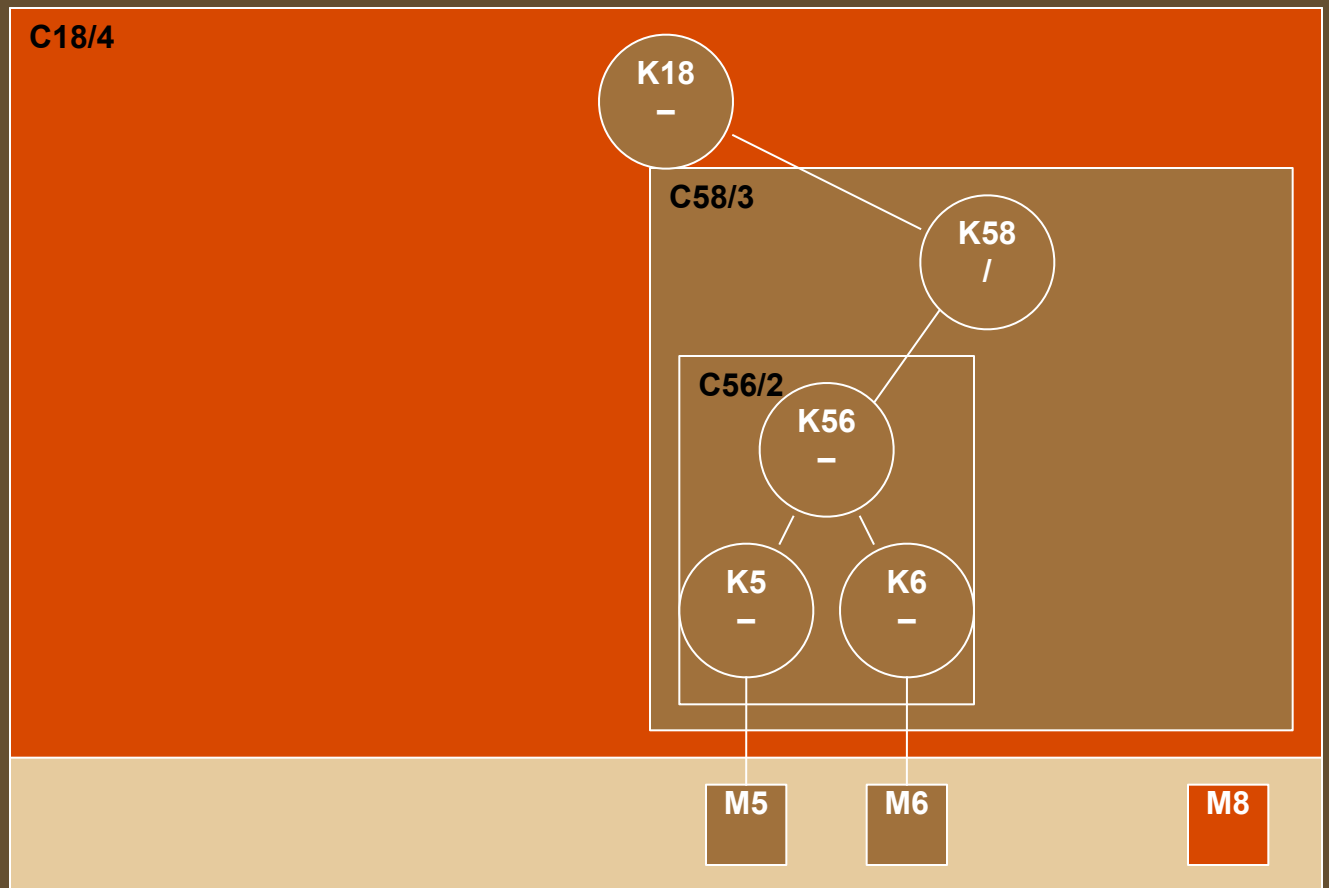


Member Joins



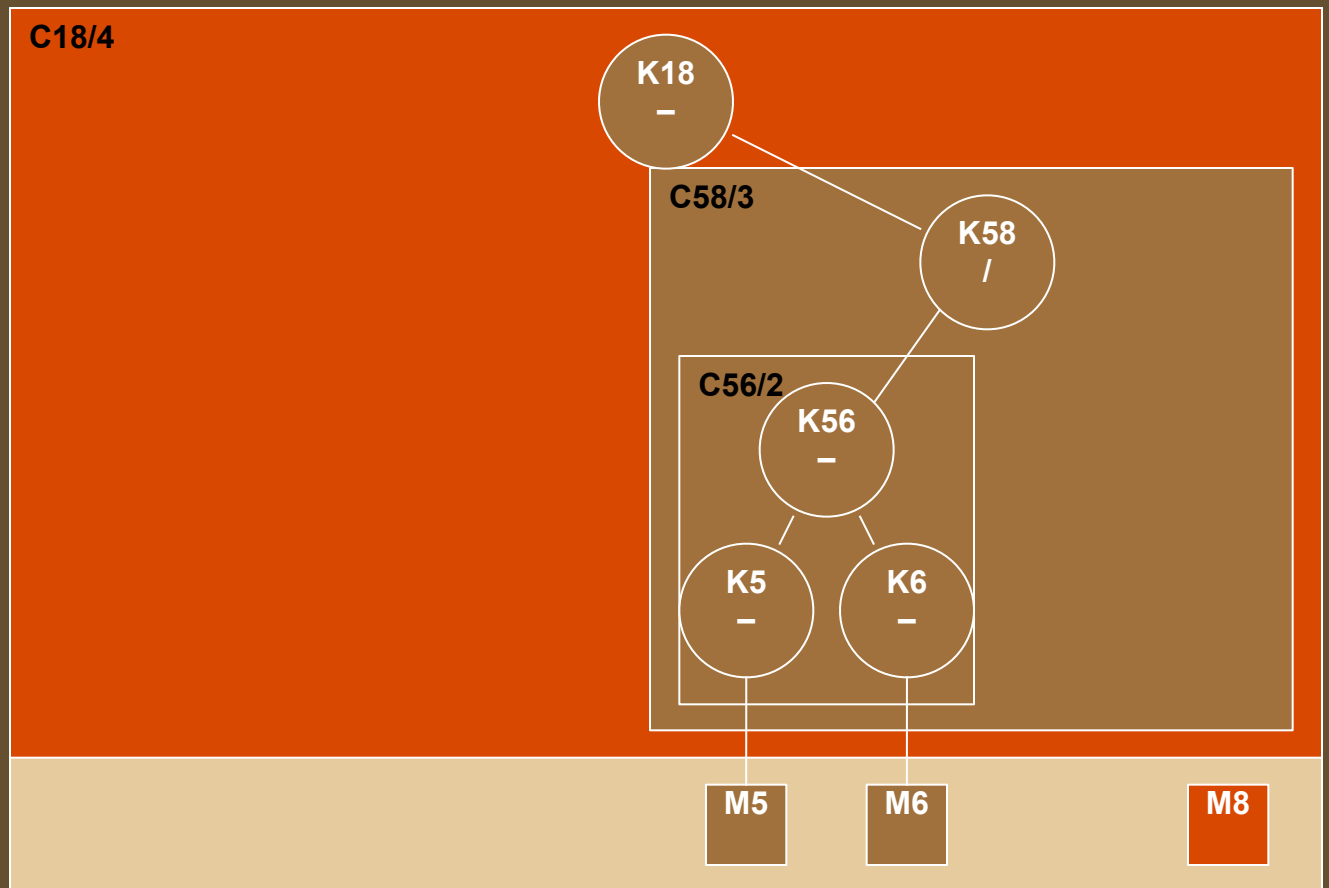
Member Joins

- Sub controller takes a random decision since the node is balanced



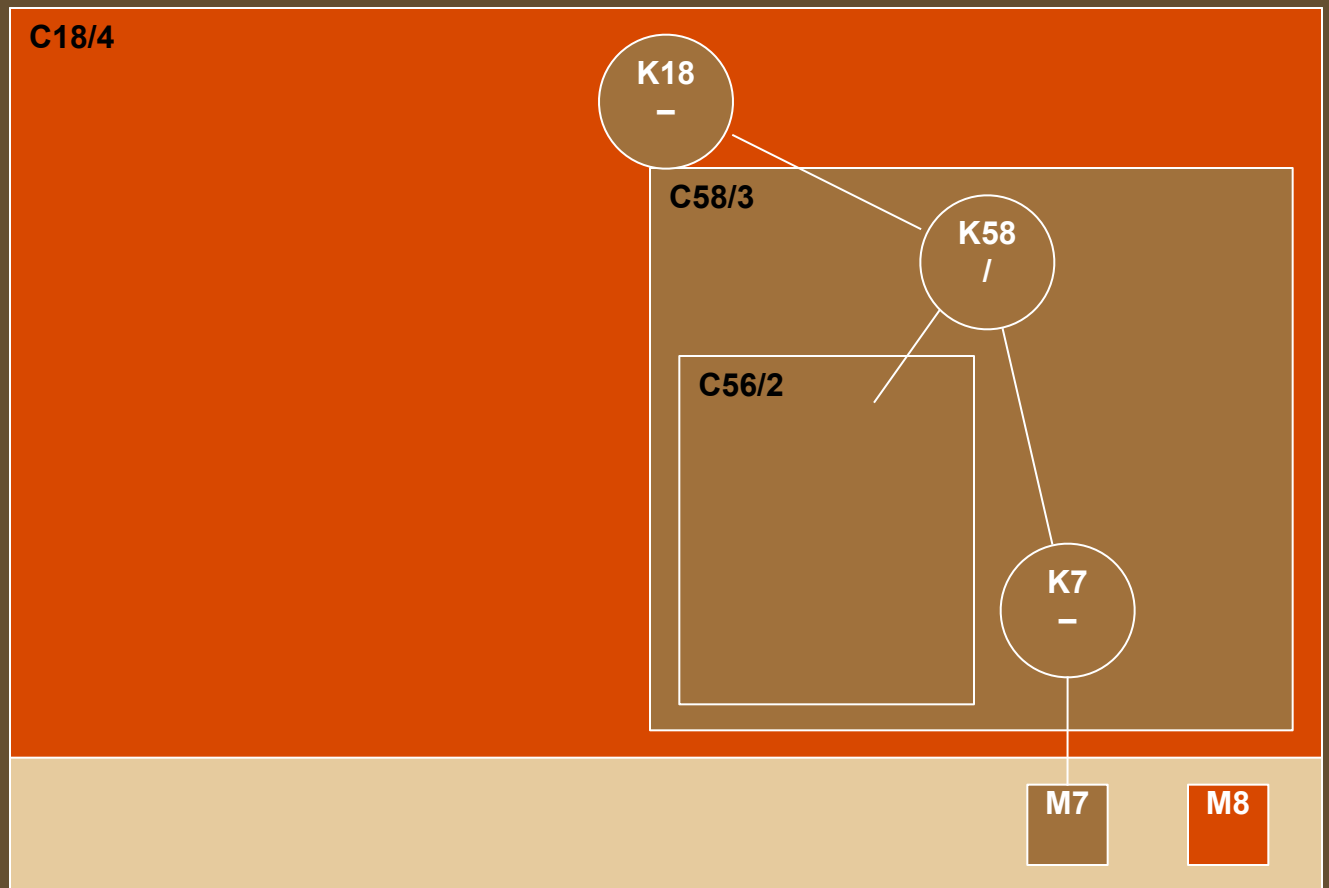
Member Joins

- Node cannot be added here since height increase is not allowed



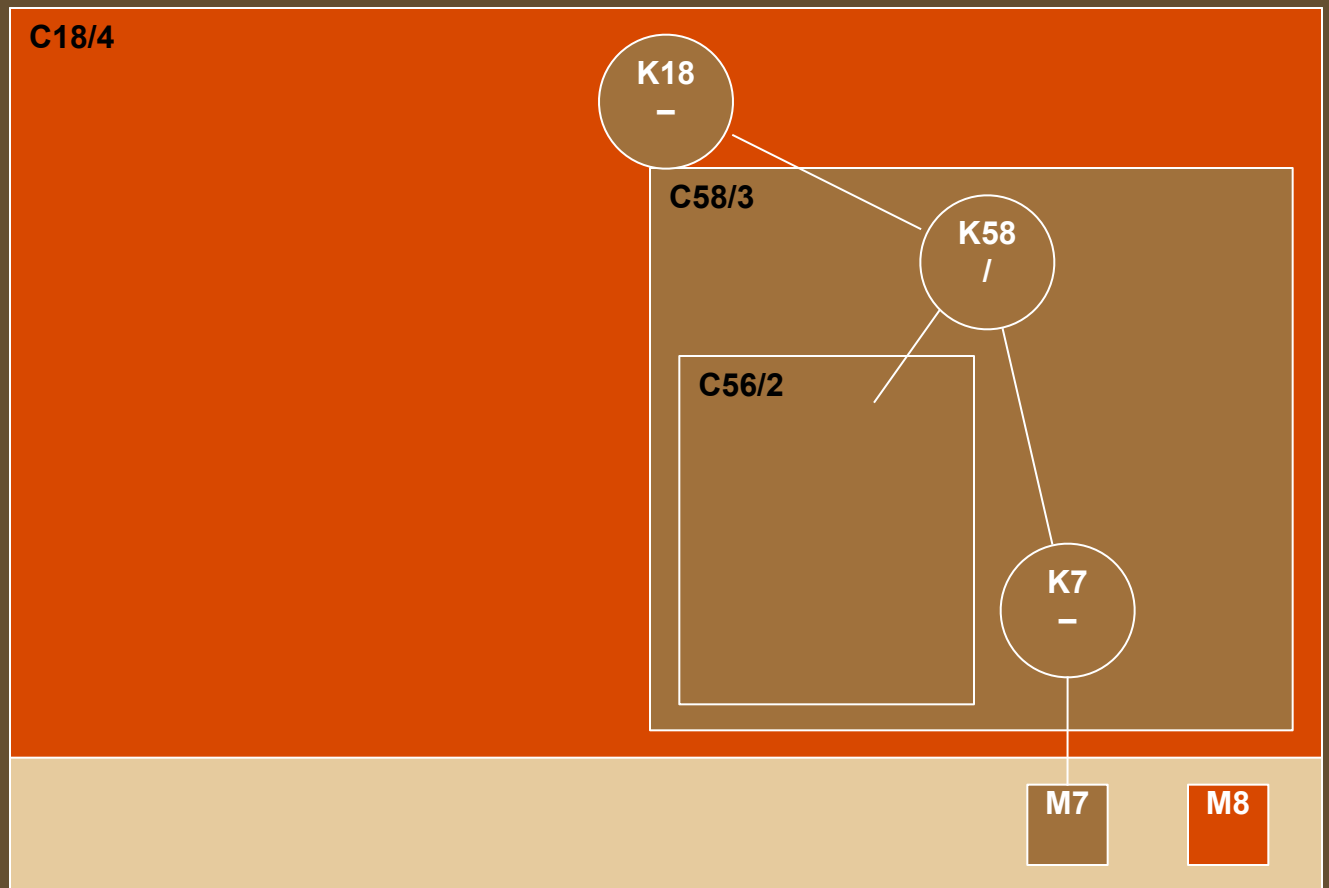
Member Joins

- Request is passed on to parent controller C58



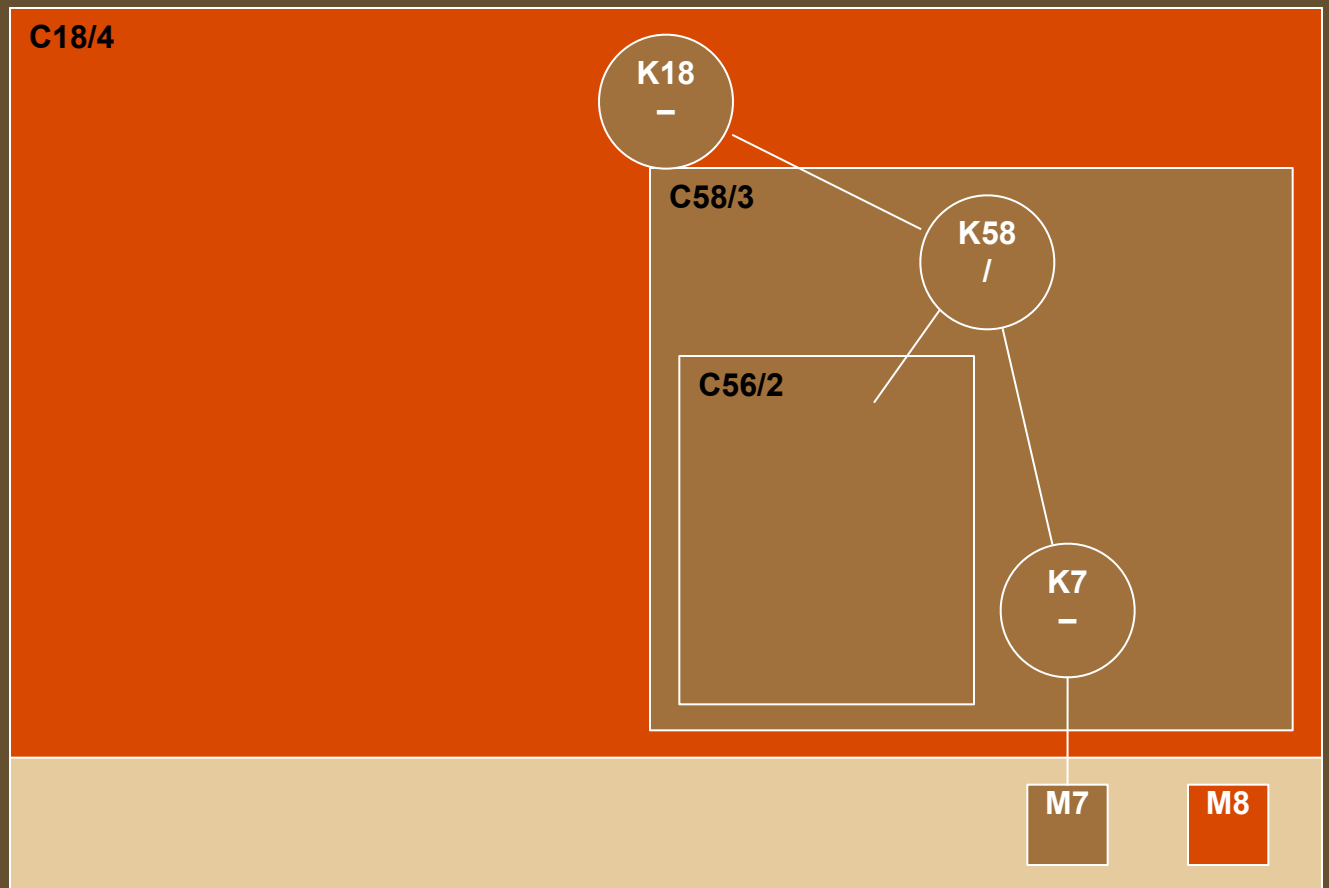
Member Joins

- Controller must add the new member on right side with height increase allowed



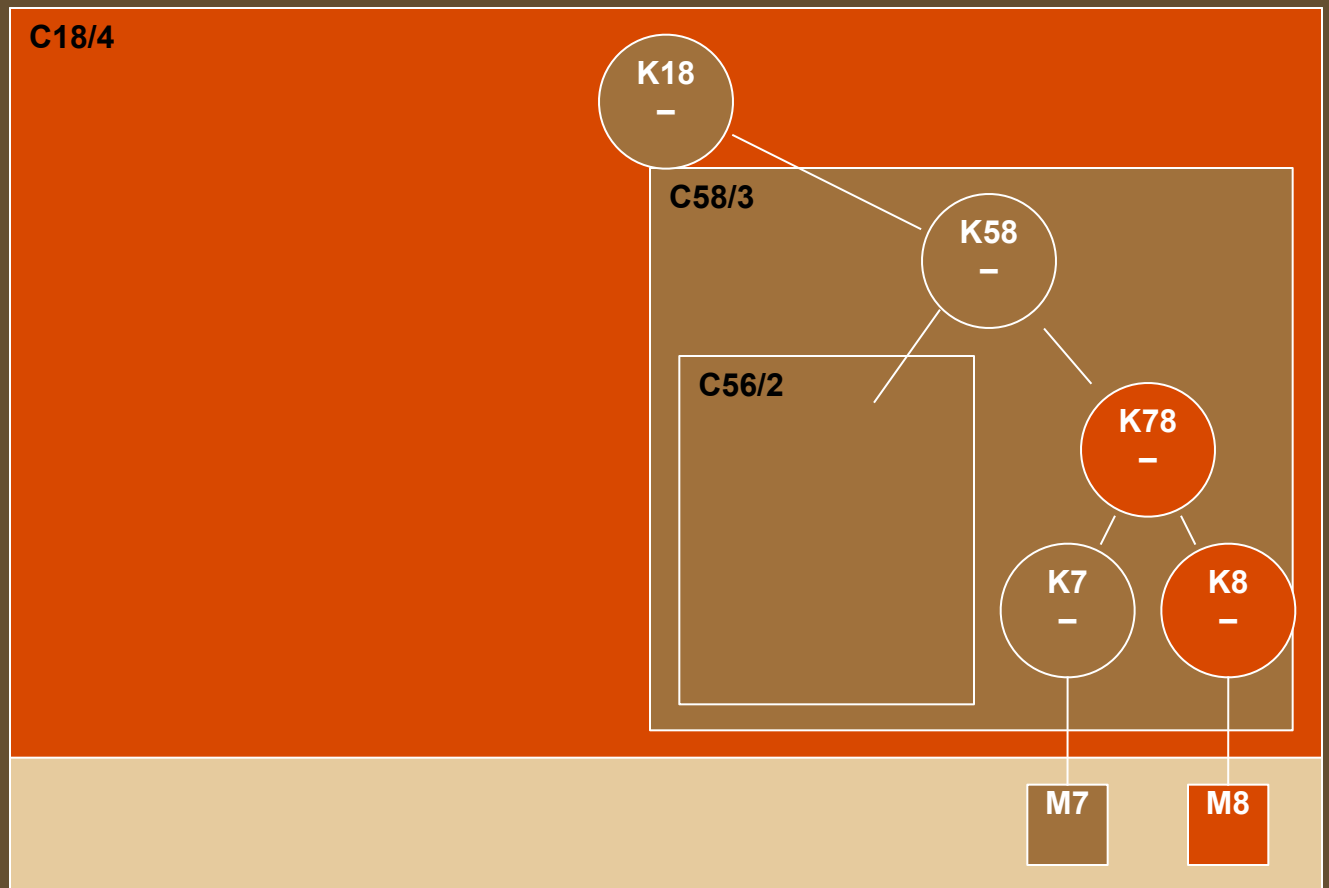
Member Joins

- Leaf node K7 must be split to accommodate for the new member



Member Joins

- Leaf node K7 must be split to accommodate for the new member

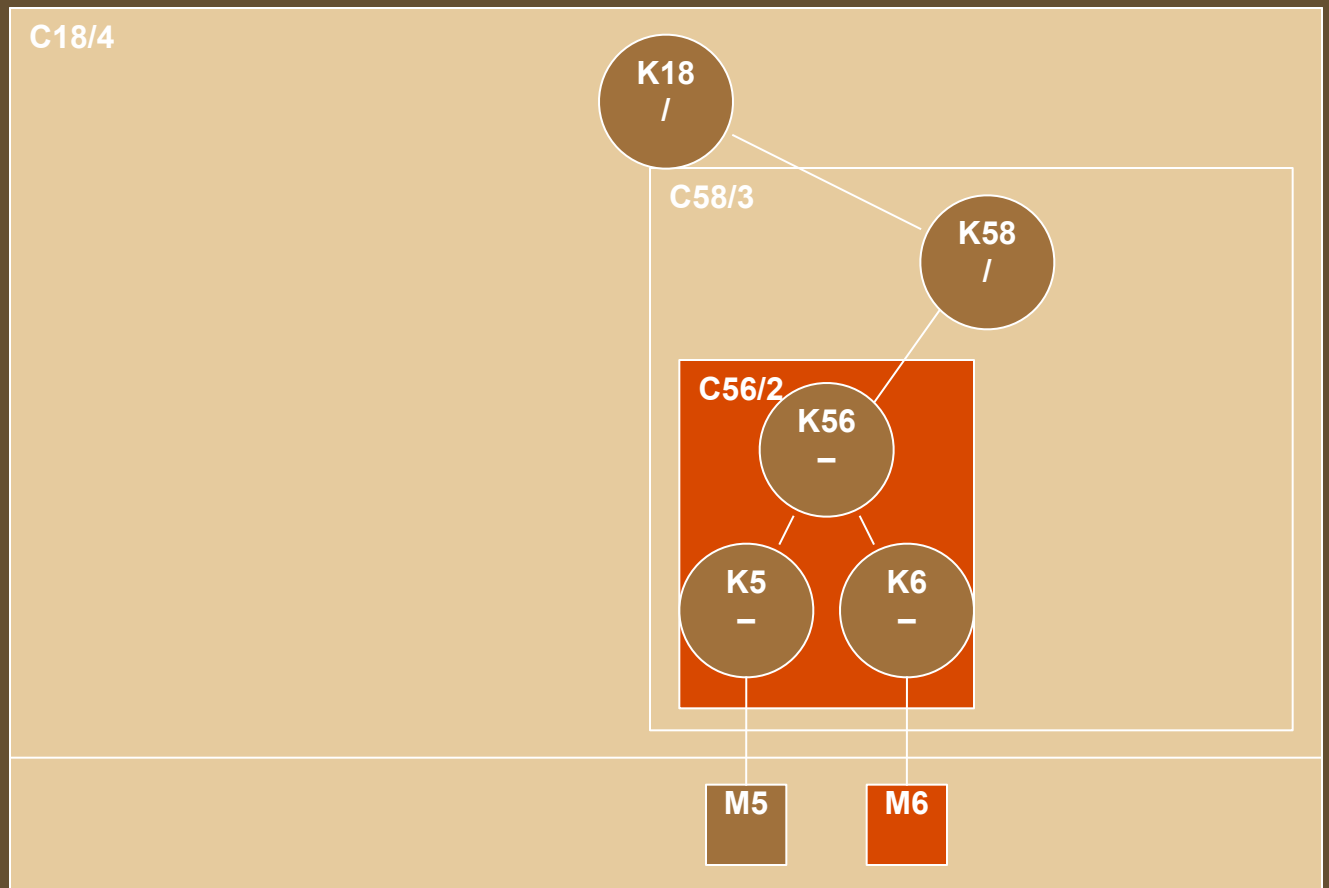


Member Joins

- To balance load the root controller forwards join requests to different controllers
- Height increase is only allowed when moving in a smaller sub-tree or when starting from root
- If insertion without height increase is impossible request is forwarded to parent controller
- Parent controller repeats the same algorithm except that it does not send the member back to the same controller

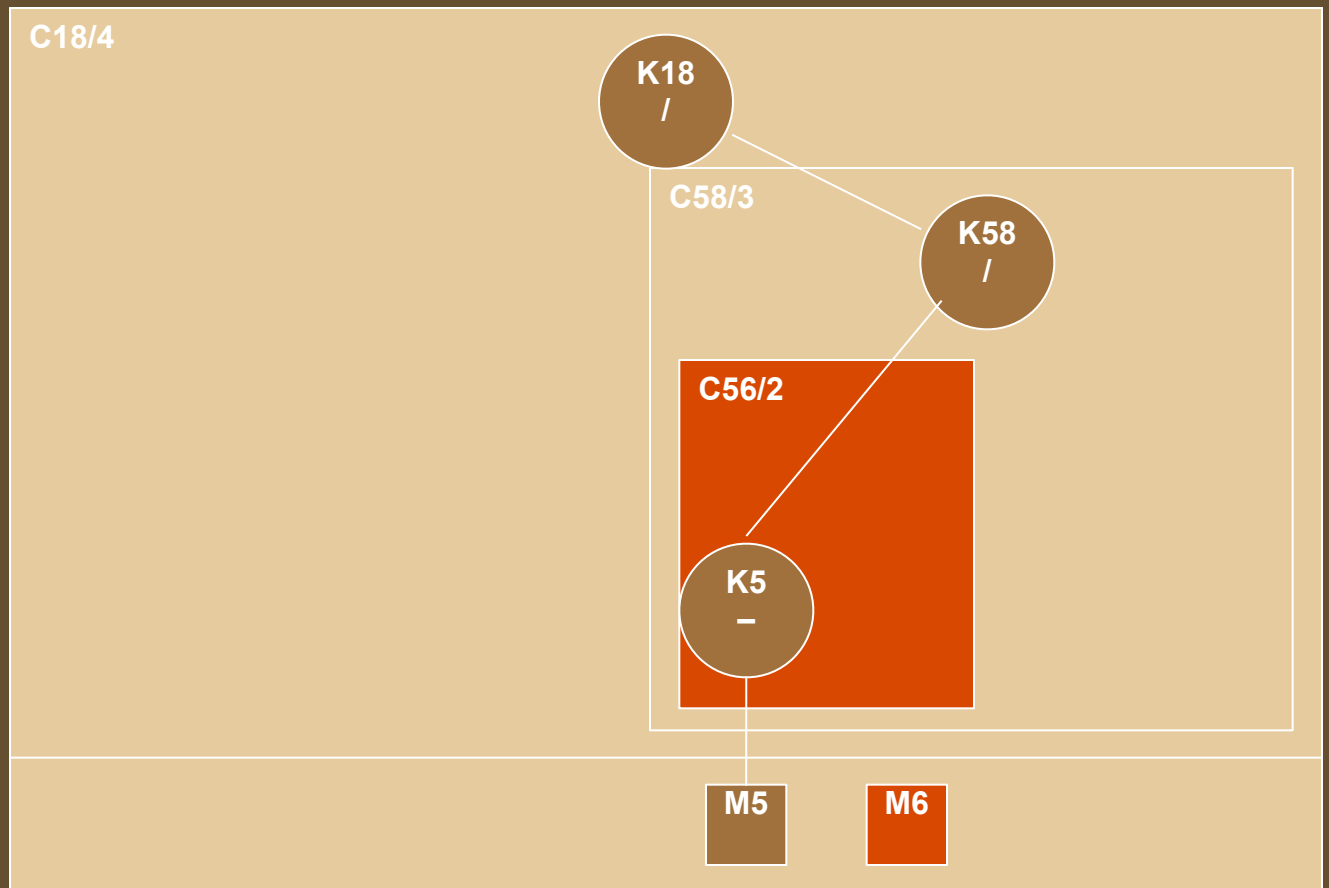
Member Leaves

- M6 wants to leave the group and informs its parent controller C56



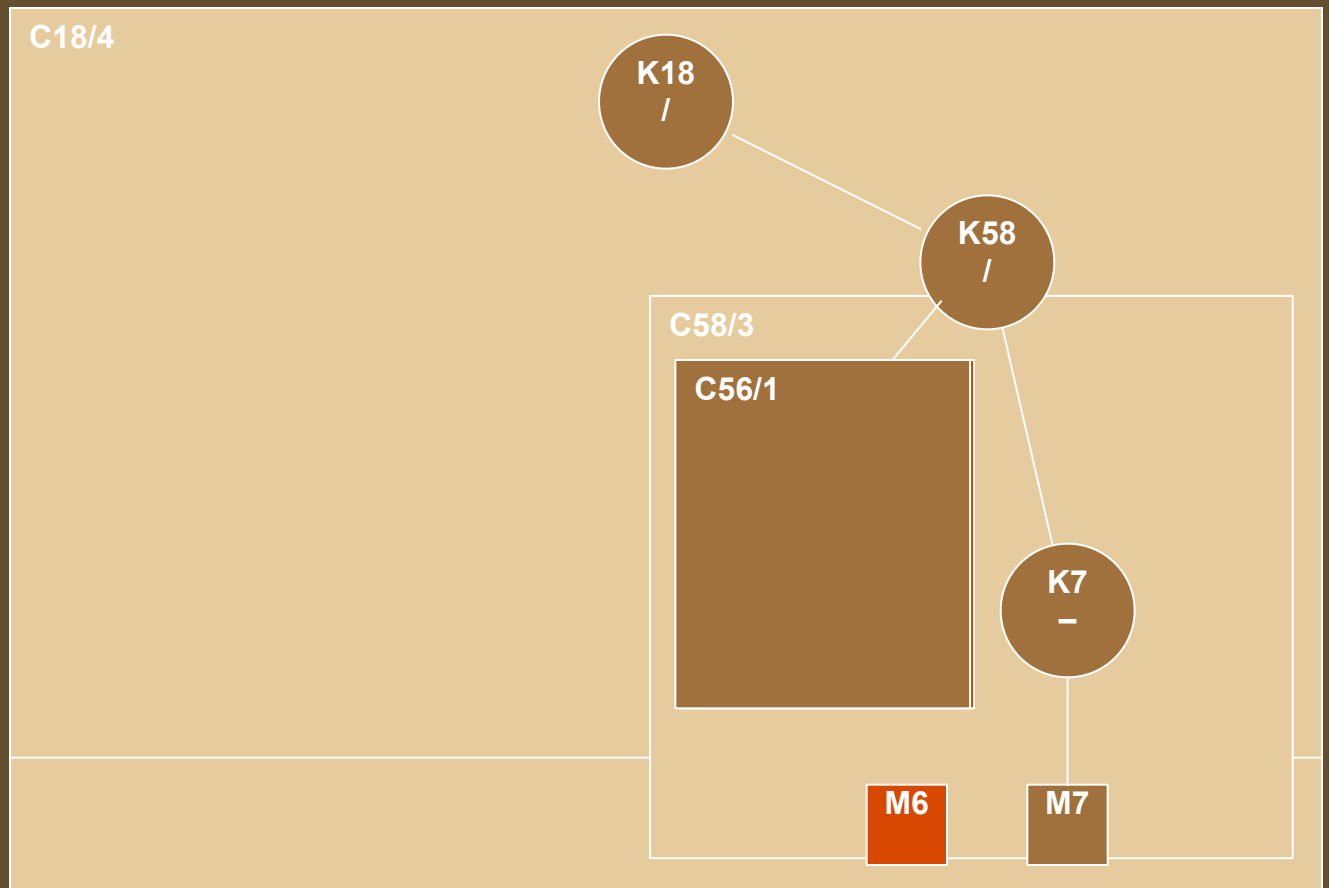
Member Leaves

- Controller merges nodes to remove M6 from the tree



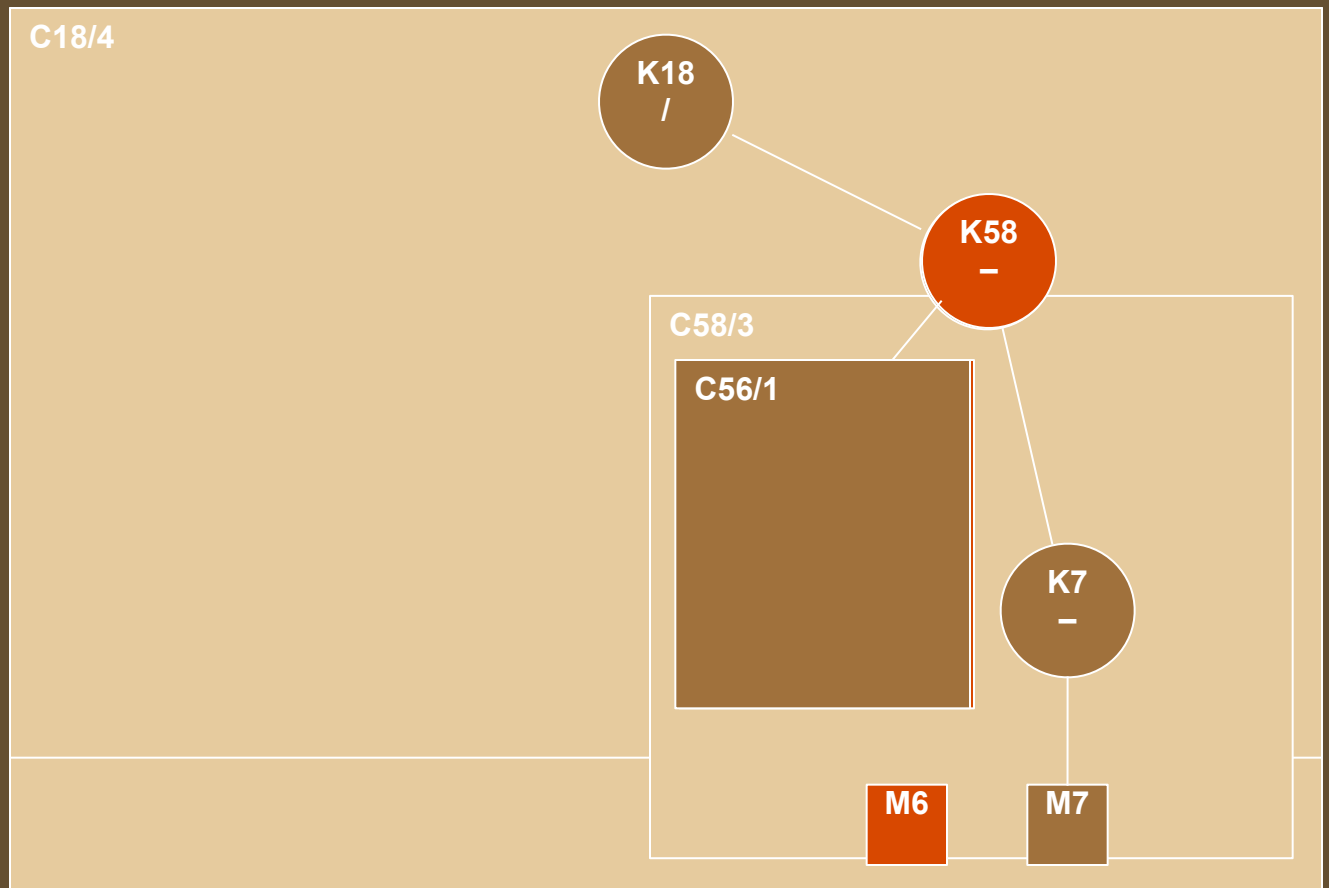
Member Leaves

- Sub controller height is changed so parent controller is informed



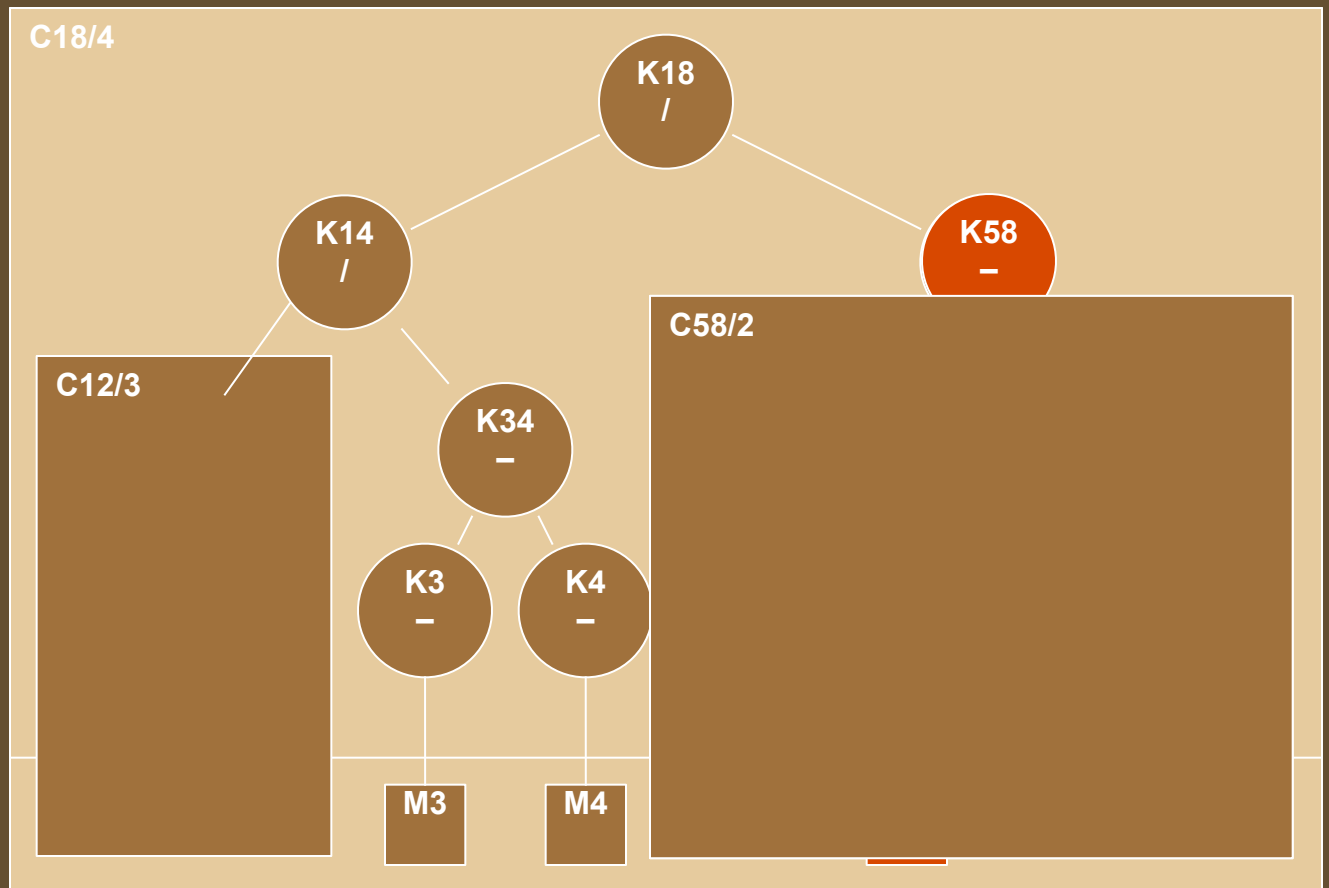
Member Leaves

- Parent adjusts the balance of its nodes



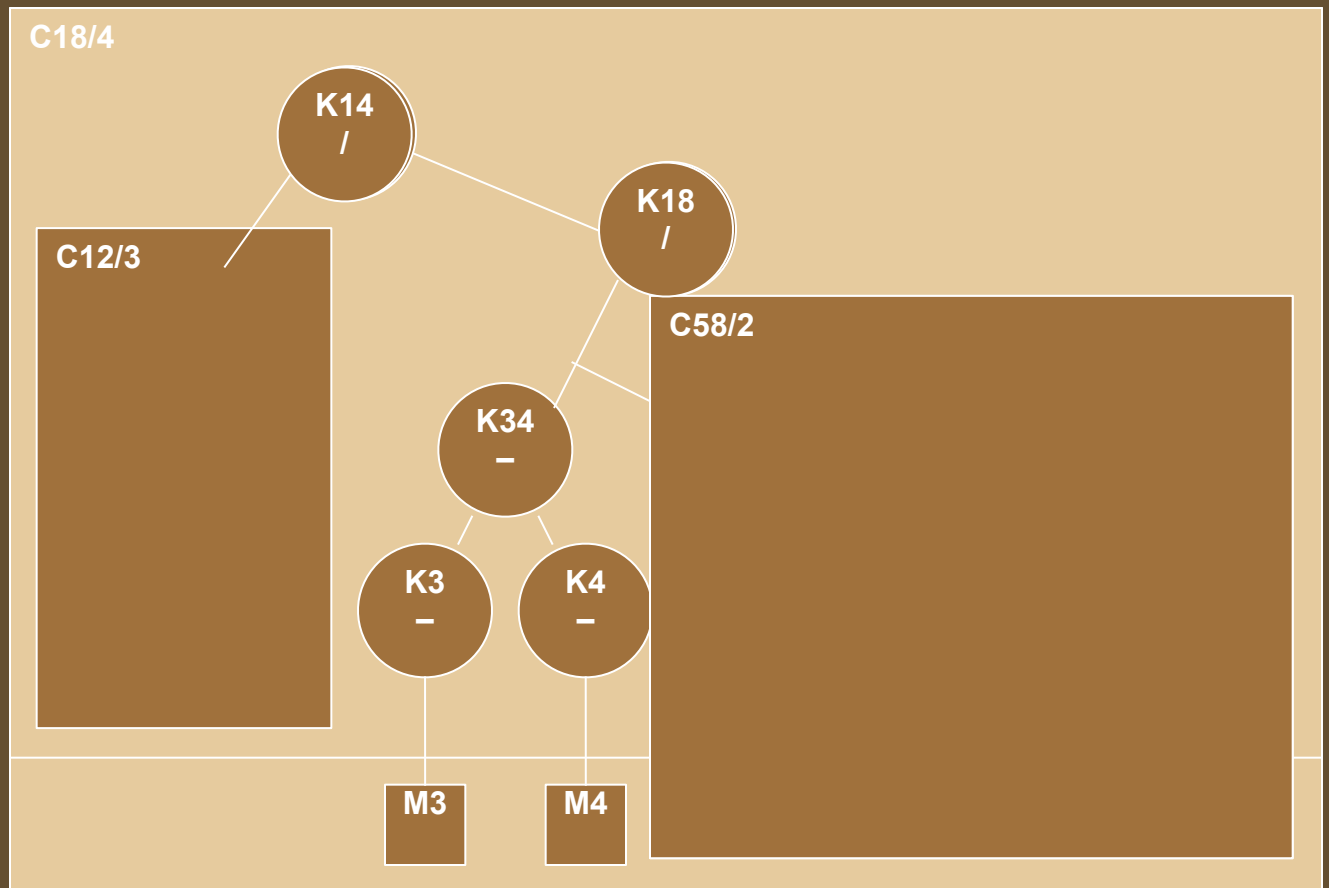
Member Leaves

- Height of controller is changed so the root controller is informed



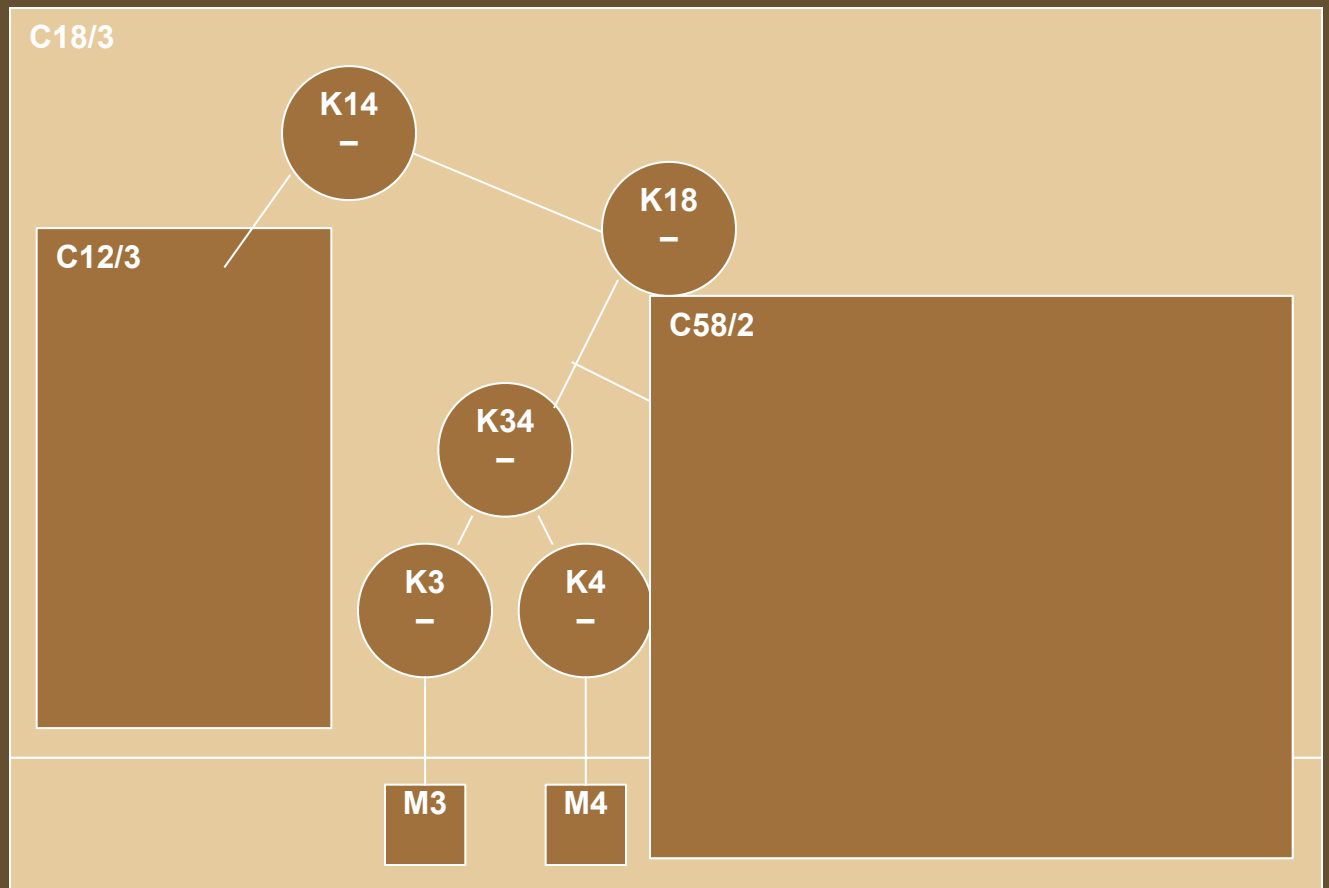
Member Leaves

- Smaller sub tree reduced in height so the root controller must undergo a rotation



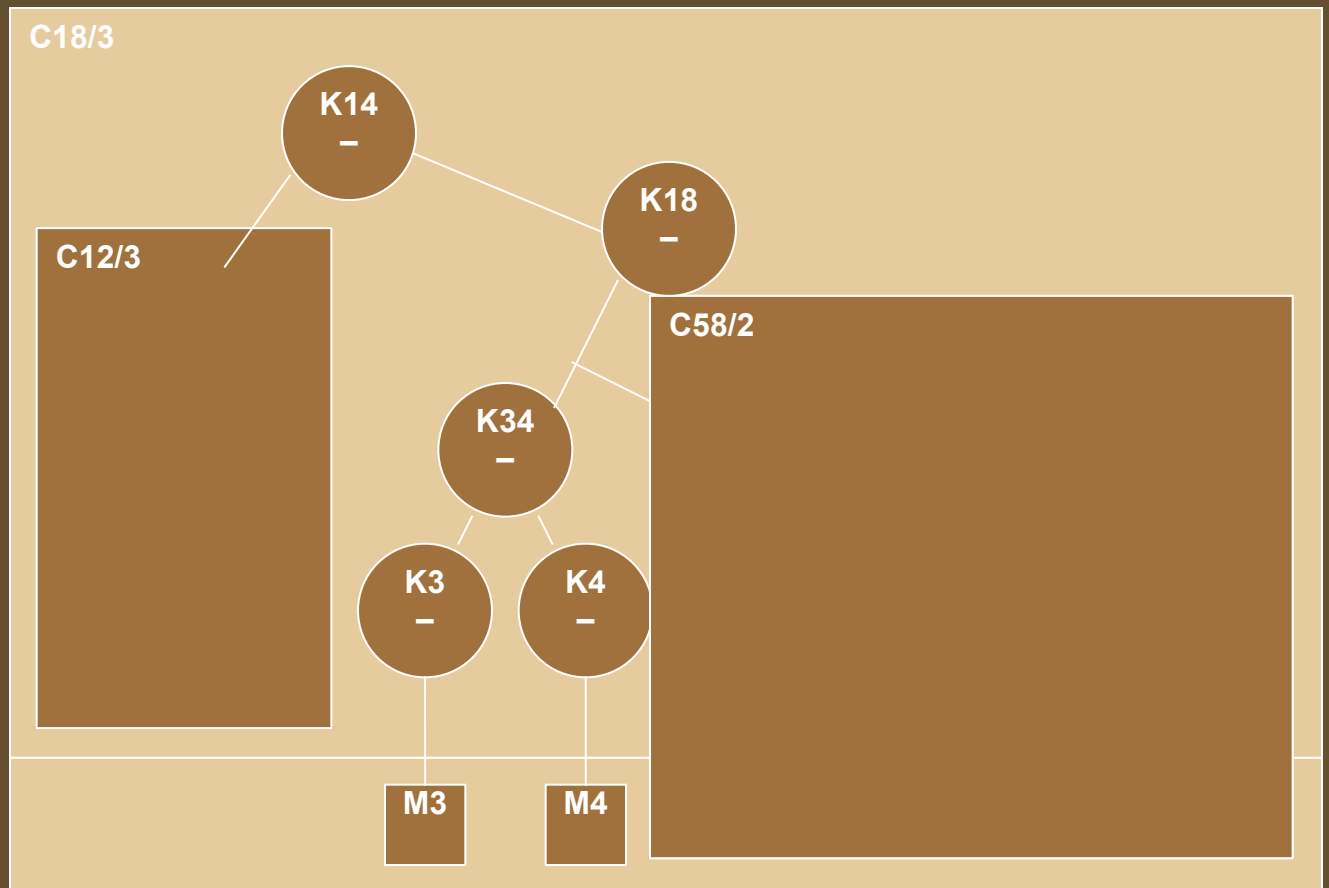
Member Leaves

- Balance of rotated nodes are adjusted



Member Leaves

- All keys on leaving member's path to the root are updated



Member Leaves

- Member informs parent controller
- Parent controller removes node from tree and adjusts nodes' balances
- If there is a difference of two in heights of sub trees a rotation is done
- If due to rotations height of total tree decreases the grandparent is informed which may itself need to go into rotations

Analysis

- Frequency of Rotations
- Analytic Comparison
- Conclusion

Frequency of Rotations

Group size	Rotations			
	Zero	One	Two	Three
10000	7692	2213	93	2
100000	77441	20535	2022	2
1000000	759008	230774	10142	76

One rotation causes 2 extra keys to be sent
0.24 rotations per leave or 0.48 keys per leave

Analytic Comparison

	LKH	Proposed LKH
Multicast message size for key Update	(2d-1)K	(2d-1+c +0.48)K
Storage at controller	(2n-1)K	(2n-1+C)K Distributed
Key update processing at Controller	(2d-1)E	(2d-1+C)E Distributed

d – depth of the tree
n – number of group members
c – number of controller in the path to the root
C – total number of controllers
K – key size
E – cost of encryption

Conclusion

- Key update increased from 2d-1 keys to 2d-1+c+0.48 keys
- However d is now guaranteed to be log n
- The overhead of balancing (0.48K) is less than overhead of height increase (2K)
- Storage and processing at controllers increases by factor of C keys but is distributed among all controllers

References

- W. Stallings; “Cryptography and Network Security – Principles and Practice”, 2nd Edition; Prentice Hall 1998
- RFC 2401, Security Architecture for IP, November 1998, S. Kent
- RFC 2406, IP Encapsulating Security Payload (ESP), November 1998, S. Kent
- RFC 2402, IP Authentication Header, November 1998, S. Kent
- Implementation and deployment of IPv6 Multicasting Jinmei T. Toshiba Corporation
- TCP/IP Blueprints Burk R., Bligh M., and Lee T. Techmedia 1998
- RFC 3376, Internet Group Management Protocol, Version 3. B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan. October 2002
- Survey of Different Multicast Routing Protocols by M. Capan, http://cn.carnet.hr/materijali/1998/980518mipro/mcapan.html
- Deploying IP Multicast in the Enterprise Thomas A. Maufer, Prentice Hall, 1998

- MBONE: Multicasting Tomorrow's Internet, A book about the multicasting backbone and the future of multimedia on the Internet Chapter 3: The MBONE and Multicasting
- IP Multicast Security: Issues and Directions by Hardjono T. and Tsudik G.
- Deployment Issues for the IP Multicast Service and Architecture
- Key Management for Multicast: Issues and Architectures, RFC 2627, D. Wallner et al. June 1999
- A taxonomy of multicast security issues <draft-irtf-smug-taxonomy-01.txt>, R. Canetti, April 1999
- Secure IP Multicast: Problem areas, Framework, and Building Blocks <draft-irtf-smug-framework-01.txt>, Thomas Hardjono, March 2001
- A Survey of Key Management for Secure Group Communication, Rafaeli S. and Hutchison D., ACM Computing Surveys, September 2003
- A Survey of Key Management for Secure Group Communication, Rafaeli S. and Hutchison D.
- Batch Rekeying for Secure Group Communications, Xiaozhou Steve Li, May 2001
- RFC 2093, Group Key Management Protocol (GKMP) Specification,

- RFC 2094, Group Key Management Protocol (GKMP) Architecture, July 1997, H. Harney
- A. Ballardie A., “Scalable Multicast Key Distribution”, RFC 1949, May 1996
- Secure Group Communications Using Key Graphs, Chung Kei Wong, IEEE/ACM TRANSACTIONS ON NETWORKING. VOL. 8, NO. 1, February 2000
- Secure Group Communication using Key Graphs, Wong K. et al., February 2000, IEEE/ACM Transactions on Networking
- D. A. McGrew and A. T. Sherman. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. Technical Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998
- Rafaeli S. et al., *“An efficient one-way function tree implementation for group key management”*
- Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization, February 1999, D. Balenson et al., IRTF Draft
- EHBT: An efficient protocol for group key management, Rafaeli S.
- ELK, a New Protocol for Efficient Large-Group Key Distribution, Perrig A. et al.
- Key Management for Secure Internet Multicast using Boolean

- Performance Optimizations for Group Key Management Schemes for Secure Multicast, Zhu S. et al.
- Optimized Group Rekey for Group Communication Systems, Rodeh O.
- A Decentralised Architecture for Group Key Management, Rafaeli S., September 2000
- Reliable Group Rekeying: A Performance Analysis, Yang R. et al.
- Group Rekeying with Limited Unicast Recovery, Brian X.
- A Scalable and Reliable Key Distribution Protocol for Multicast Group Rekeying, Setia S.
- Keystone: A Group Key Management Service, Wong C. and Lam S., Proceedings International Conference on Telecommunications, May 2000
- The VersaKey Framework: Versatile Group Key Management, Waldvogel M. et al.