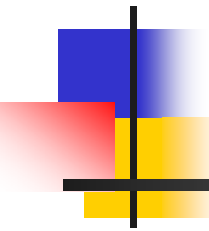
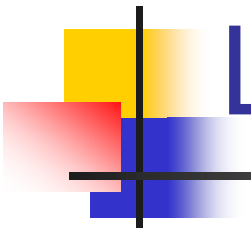


Secure Multicast

Guevara Noubir

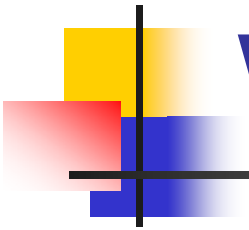
noubir@ccs.neu.edu

CCIS - NEU



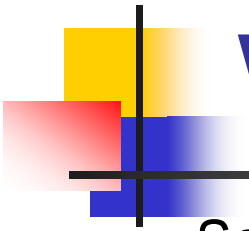
Lecture Outline

- Introduction to multicast
- Multicast on the internet
  - Multicast over ethernet
  - Routing protocols for IP multicast
  - Mbone
- Securing multicast groups



What is Multicast?

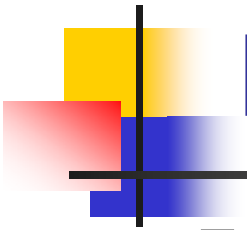
- Multicast is a communication paradigm
  - 1 source, multiple destination
- Applications:
  - **bulk-data distribution to subscribers**
    - **(e.g., newspaper, software, and video tapes distribution),**
  - **connection-time-based charging data distribution**
    - **(e.g., financial data, stock market information, and news tickets broadcasting),**
  - **streaming (e.g., video/audio real-time distribution),**
  - **push applications, web-casting, CDN,**
  - **distance learning, conferencing, collaborative work, distributed simulation, and interactive games.**



Why Multicasting?

- Several applications need efficient means to transmit data to multiple destinations with:
  - less bandwidth
  - higher throughput
  - higher reliability
  - lower delay

- Classifications
  - Interactive Real-time, Reliable Multicast apps, Streaming apps
  - Data dissemination, Transactions, Large Scale Virtual Environments



Ethernet Multicast

- Ethernet is a broadcast medium
  - Every frame can potentially be seen by every host
- Ethernet cards have a unique Ethernet address
- Broadcast address:
  - ff:ff:ff:ff:ff:ff
- Ethernet Multicast address range for IP:
  - 01:00:5e:00:00:00 -to- 01:00:5e:7f:ff:ff

Mapping IP Multicast onto Ethernet Multicast

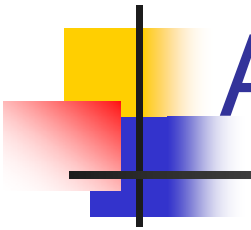
- IP Multicast (class D IP address):
  - Class D: 224.x.x.x-239.x.x.x (in HEX: Ex.xx.xx.xx): 28 bits
  - No further structure (like Class A, B, or C)
  - Not addresses but identifiers of groups
  - Some of them are assigned by the IANA to *permanent host groups*

- Mapping a class D IP adr. into an Ethernet multicast adr.
  - The least 23 bits of the Class D address are inserted into the 23 bits of ethernet multicast address
  - Many to one mapping: 5 bits are not used
  - More filtering has to be done at IP level

IP Mutlicast: Problems to Solve

- Build on top of the existing Internet and take into account group communication constraints
  - Manage groups
  - Create and maintain multicast routes
  - Efficient control mechanisms:
    - reliability, flow control, time constraints

Shortest Path Tree Routing Algorithm

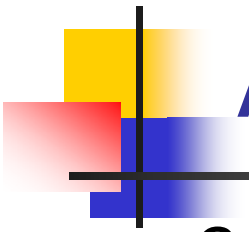


- Apply point-to-point shortest path for all the receivers
- Multiple sources compute different trees
- For dynamic networks: 2 techniques to gather info
  - Distance vector algorithm
    - Each router sends to its neighbors its distance to the sender (called vector distance)
    - After receiving the vector distance from its neighbors, each router computes its own vector distance (minimum(received_vectors)+cost-to-neighbor)
  - Link state algorithm
    - Network connectivity information is broadcast to all routers
    - Every router has a complete knowledge of the network state
    - Every router centrally computes (using Dijkstra's algorithm) the shortest path to the sender

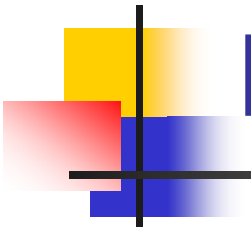
Minimum Cost Tree Routing Algorithm

- Goal: minimize the overall cost of the multicast tree
- Minimum Spanning Tree:
  - Minimum cost tree which spans all nodes (Prim-Dijkstra's algorithm: add nearest members one by one to the tree)
  - Example:
- Minimum Cost Steiner Tree:
  - Minimum cost tree which spans at least all the group members
  - This problem in NP-complete: we don't have an algorithm that can solve it in polynomial time of the size of the graph (stays NP-complete when link cost = 1, planar graph, bipartite graph)
  - Heuristics exist for approximating the minimum Steiner tree

Constrained Tree Routing Algorithm



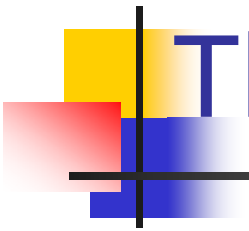
- Goal: minimize both the distance between the sender and the receiver (delay) and the overall tree cost (bandwidth)
- Reason: real applications have constraints on delay/cost.
- Heuristics:
  - e.g., [Kompella, Pasquale, Polyzos 93: IEEE/ACM Trans. Net.]



Practical Systems

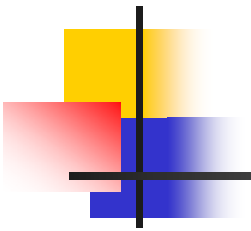
- DVMRP: distributed implementation of Shortest Path (Bellman-Ford Alg.)
- MOSPF: Shortest Path algorithm (link-state Dijkstra's Alg.)
- CBT: center-based tree
- PIM (sparse mode): Center-based tree + Bellman-Ford

Multicast Routing Protocols: The Evolution



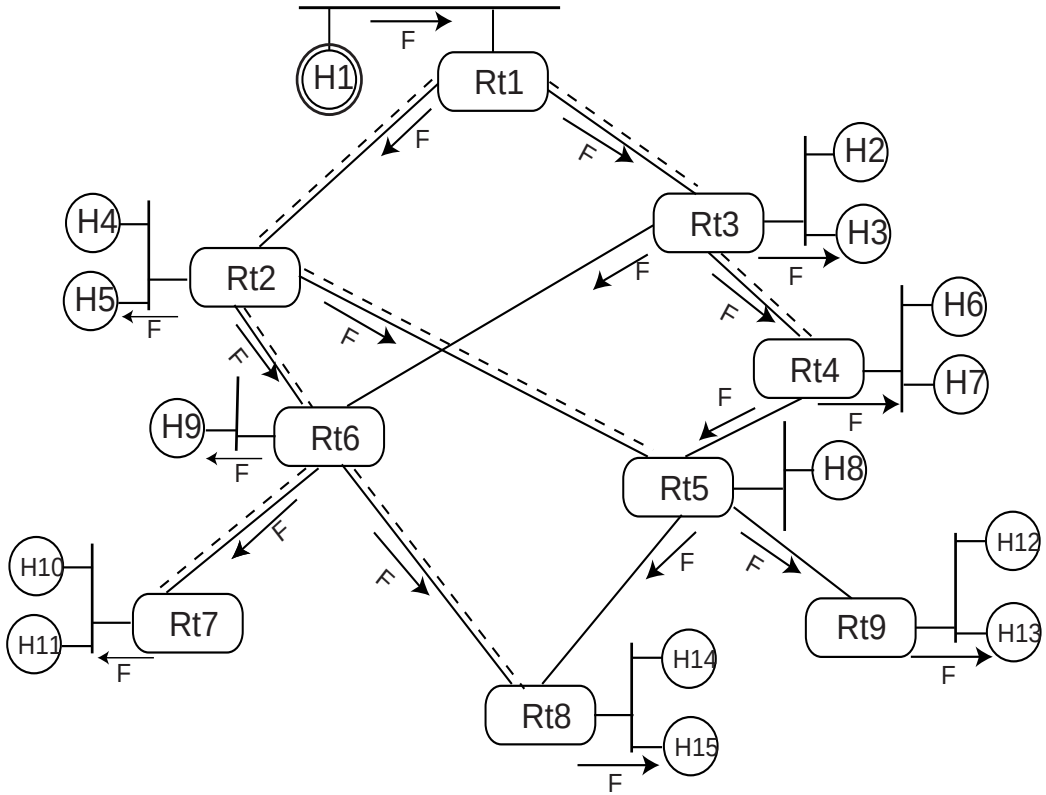
- Reverse Path Forwarding (RPF)
- Internet Group Management Protocol
- Truncated Broadcasting
- Distance Vector Multicast Routing Protocol (DVMRP)
- Multicast extensions to Open Shortest Path First (MOSPF)
- Protocol Independent Multicast (PIM)
- Core Based Tree (CBT)
- Ordered Core Based Tree (OCBT)
- Hierarchical DVMRP (HDVMRP)
- Hierarchical PIM (HPIM)
- Border Gateway Multicast Protocol (BGMP)

Reverse Path Forwarding [Dalal, Metcalfe 78]



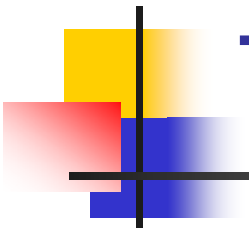
- If a router receives a packet on the interface that leads to the multicast sender, he forwards the packet on the other interfaces. Otherwise, he drops the packet
- This protocol achieves broadcasting, but not multicasting
- We need a mechanism to know where are the members of the group

Illustration of RPF



Internet Group Management Protocol [RFC1112, 1989-97]

- IGMP router periodically broadcasts a *Host-Membership Query* on its subnet
- If there is a host subscribing to the group, the host schedules a random timer to send an *IGMP Host-Membership Report*
- When the timer expires the *IGMP H-M Report* is multicasted. The purpose of this report is:
  - The other members of the group in the same subnet cancel their timer
  - The router knows that there is a member on its subnet listening to a given group



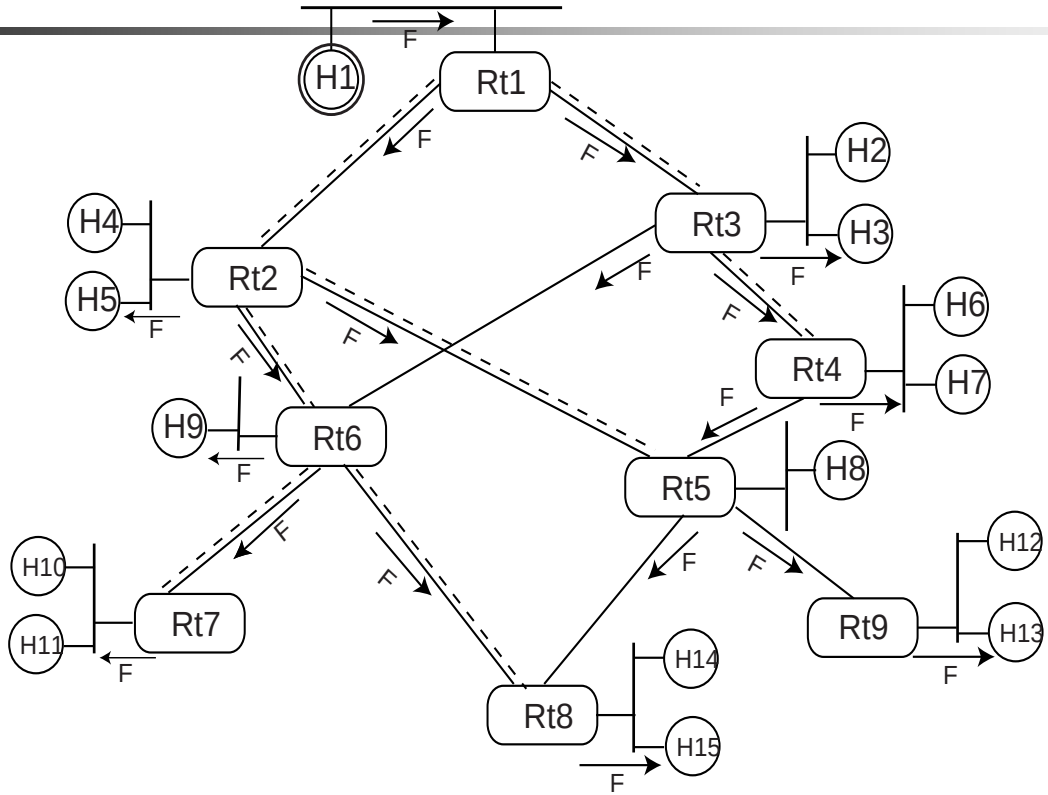
Truncated Broadcasting

- Uses the group membership information to decide if the packets will be broadcast on the leaf subnet
- Reduces the traffic in the leaf subnet
- Does not reduce the traffic in the core network

Distance Vector Multicast Routing Protocol (DVMRP): RFC1075(1988-97)

- Distance vector routing
  - Similar to RIP and extended to multicast routing
  - Extends truncated broadcast by using *pruning* and *grafting*
  - *Soft-state* protocol: pruning and flooding is periodically repeated
- Pruning:
  - On reception of a flooded packet:
    - Sends the *prune* message on the interfaces different from the *reverse shortest path*
    - If the leaf- router is not interested (no members) it sends a *prune* on reverse path
    - If a router receives a prune on all its interfaces except the reverse shortest path, it propagates the prune through the reverse shortest path
- Grafting: If a host wants to join before the next flooding:
  - a graft is forwarded upstream (RPF) to the closest router in the tree

Illustrating DVMRP



Summary of some of the problems

- Flooding/pruning:
  - good for small dense networks
  - bad in poorly populated networks
- Sender specific trees:
  - low delay
  - complex routing tables
- Shared trees:
  - small routing tables
  - traffic concentration, non-optimal delay
- Low Cost Steiner trees:
  - good overall cost
  - might be too complex to compute on the fly

Protocol Independent Multicast (PIM: 1996)

- Goals:
  - does not depend on any unicast protocol
  - optimizes traffic depending on the density of receivers in the region
  - low-latency data distribution (source-based trees instead of shared-trees)
- Modes:
  - Dense mode: flooding
  - Sparse mode: use Rendezvous Points (RPs)
- Sparse mode regions:
  - number of networks/domains with members is significantly smaller than the total number of networks/domains in the region
  - group members are widely distributed
  - overhead of flooding + pruning is high

Components of PIM

- Rendezvous Point (RP):
  - each multicast group uses one RP:
    - (SM) receivers explicitly join the group by sending a *JOIN* to the RP
    - senders unicast to the RP, which sends the packets on the shared tree
- Designated Router (DR):
  - each sender/receiver communicates with a directly connected router (PIM-Reg: Join/Prune)
  - the DR may be the IGMP querier
- Last Hop Router (LHR):
  - router directly connected to the receiver: forwards the multicast packets
  - generally: LHR = DR
- Boot Strap Router: elected router within a domain
  - constructs the set of RP and distribute it to the routers in the domain

Key Steps of PIM

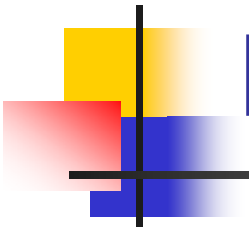
- Creating the PIM framework:
  - some routers are configured as candidate RPs (C-RPs)
  - C-RPs periodically send C-RP-Advs to the BSR
  - BSR distributes the RP-set to all the routers (Bootstrap Messages: BSM)
  - any router: RP-set + Group Address -> RP for the group
- Multicast shared tree:
  - Receiver join:
    - IGMP-report message from receiver to DR
    - DR creates an entry (*, G), DR sends a PIM Join/Prune message to RP
  - Source Join:
    - IGMP-report message from sender to DR
    - Data packets are unicast to the RP by the DR: PIM-register
    - Packets are forwarded through the shared tree (if there is no (S, G) entry: no shortest path tree)

Key Steps of PIM (*Cont'd*)

- Switching from shared tree to shortest path tree:
  - PIM starts with a shared tree (RP-tree)
  - when the traffic > TH, the receiver DR/LHR initiates the switch:
    - creates a source specific entry (S1, G)
    - sends a PIM Join/Prune to the sender through the next best hop router for S1
    - intermediate routers send a PIM Join/Prune to the sender on the shortest path
    - intermediate routers send a PIM Join/Prune to the RP if the path to the RP is different from the shortest path
- Steady state maintenance:
  - soft state protocol: periodic join/prune messages
- Data forwarding:
  - first check for a (S, G) entry: SPT, otherwise for (*, G): shared tree

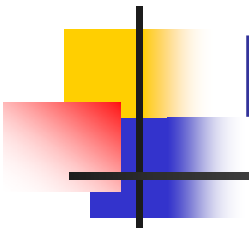
Multicast in IPv6

- Multicast address format (128 bits): FF.FlagScope.G-ID
  - Flag (4bits):
    - 0: permanently assigned group (NTP, ...)
    - 1: transient group
    - others: undefined
  - Scope (4bits):
    - limits of transmission (nodes, links, sites, organization)
  - Group-ID (112bits):
    - unique group ID
    - reserved values: 0 (never used), 1: all nodes, 2: all routers
- Group-ID is assigned using random number generators
- IGMP is incorporated inside ICMP



Multicast Backbone (Mbone)

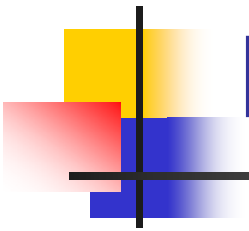
- Multicast chicken-and-egg problem:
  - multicast cannot be deployed (and fully tested) without the support of router vendors
  - router vendors would not support IP multicast before it is mature and robust
- Mbone solution:
  - connect multicast capable routers using IP tunnels
  - First IP tunnel 1988: BBN (Boston) and Stanford University
  - IEEE INFOCOM, IEEE GLOBECOM, ACM SIGCOMM over MBone
- Tunneling:
  - IP multicast packets are encapsulated into unicast packets and sent to next-hop MBone router
  - Next MBone router strip off the outer packet header:
    - multicast to its subnet (if there is any members)
    - re-encapsulate the packet and send it to the next-hop using IP tunnel



Mbone (*Cont'd*)

- Traffic level in the MBone
  - Upper limit per tunnel: 500 KBps
  - Typical conference sessions: 100-300 KBps
  - TTL (0-255) to limit the scope of sessions

- MBone tools
  - session directory (sd, sdr)
  - audio conferencing tool (vat, nevot, rat)
  - video conferencing tool (nv, ivs, vic, nevit)
  - shared whiteboard tool (wb)
  - Network text editor (nte)

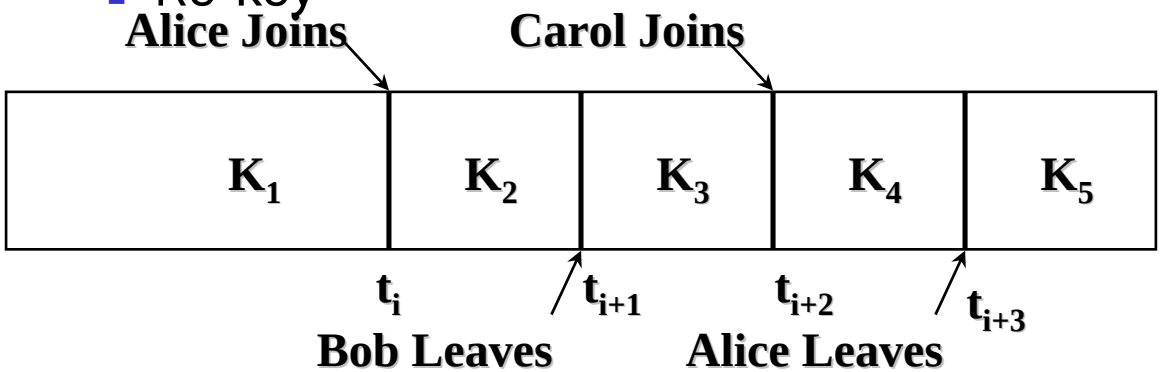


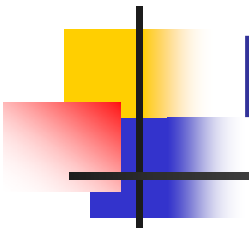
Multicast Security

- Authentication, Confidentiality, Integrity: IPsec
- Integrity protection against group members
  - If symmetric integrity algorithms are used, any group member can modify packets and compute a new integrity checksum
  - Use only asymmetric integrity algorithms, amortized signatures, hash chains
- Non-repudiation, Security audit: added at the application level

Multicast Key Management Problem

- Key distribution:
  - Static groups:
    - Key establishment
    - Re-key
  - Dynamic groups:
    - Group membership change (e.g., Join and Leave)
    - Re-key





Early Solutions

- Classical:

  Does not change the key on membership change

- Group Key Management Protocol (GKMP Internet draft) [1994]:

  Does not address the scalability issue

- Scalable Multicast Key Distribution (SMKD RFC) [1996]:

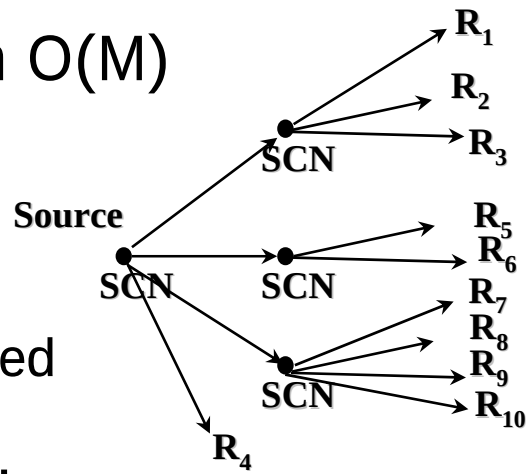
  Does only initial distribution (no key change on membership change)

Early Solutions (Con't)

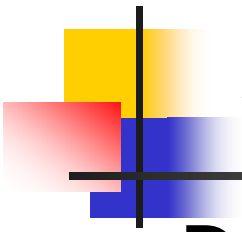
- Iolus [1997]:

  Achieves a scalable key change on group membership change

  - reduces the key update message from O(M) to O(M/N)
    - may still be very high for large groups
  - requires several trusted authorities
    - more vulnerable to attacks since any trusted authority can be attacked
  - Is not adapted to multicast over satellite
    - requires physically distributed hierarchy of group controllers



Other Group Key Distribution Schemes



- Diffie-Hellman key exchange extension [1996]:
  - M round protocol
  - Exponentiation of big numbers
- Chinese Remainder theorem secure locks [1989]
- Polynomial interpolation [1995]:

Centralized Key-Management Based on Partitioning

- Concept of the proposed scheme:
  - Use a set *KS* of keys
  - Each group member has a unique key shared with the GC
  - Each group member M_i has a subset of keys KS_i
  - Every member M_j ($\neq M_i$) has at least one key that M_i does not have

$$\forall j \neq i; KS_j \cap (KS - KS_i) \neq \varnothing$$

- Assumption no collusions
- Example: 20 members, 6 keys

		Group Members														**Codes**					
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
KS	K_1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	K_2	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0
	K_3	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0
	K_4	0	1	0	0	1	0	0	1	1	0	1	0	0	1	1	0	1	1	0	1
	K_5	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1
	K_6	0	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	1	1	1

Robustness of the PBKM

- The PBKM does not resist to multi-user attacks:
  - Alice Joins the group => new group key K_1
  - Bob joins the group => new group key K_2
  - Alice leaves the group => new group key K_3
  - Bob leaves the group => new group key K_4
  - Bob sends K_3 to Alice
  - Alice recovers K_4 from K_3 and KS_{Alice}

- Conclusion: two members can leave a group and manage to still have access to the group key

Logical Key Tree

- The problem of the partitioning scheme is that a member that leaves the group keeps some valid keys
- The Logical Key Tree changes all the keys of a leaving member
- The Logical Key Tree requires:
  - only 2*log(|M|)-1 messages when a member leaves the group
  - storing 2*|M|-1 keys.
- Is centralized at the group controller
- Adequate for mutlicasting over satellite links
- Independently discovered by Caronni et al., Noubir, Wong et al., 1998.

Example

- M_4 leaves the group: K_{10}, K_1, K have to be changed

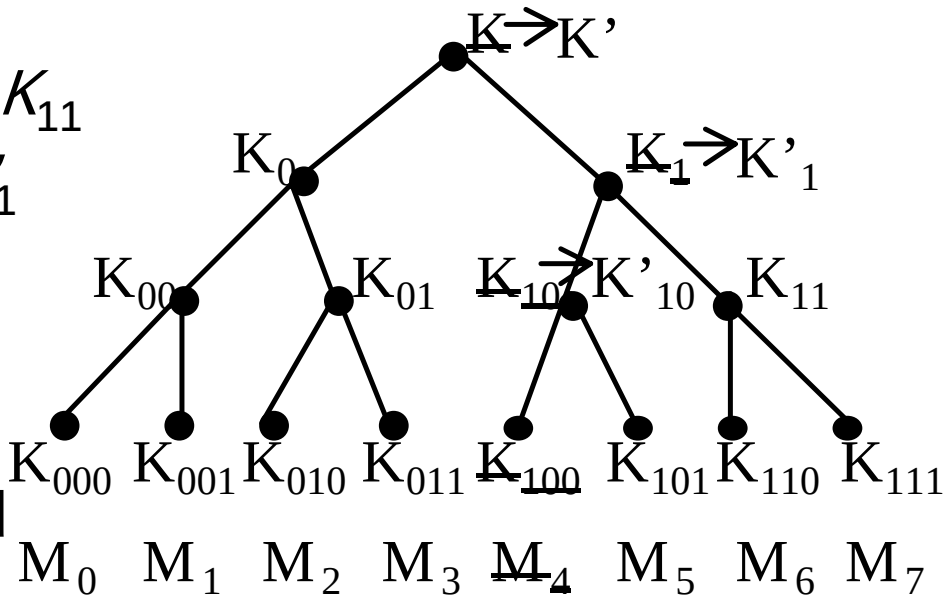
  GC sends:

  - K'_{10} encrypted using K_{101}
  - K'_1 encrypted using K'_{10}, K_{11}
  - K' encrypted using K_0, K'_1

- M_4 joins:

  GC sends:

  - K'' encrypted using K'
  - K'_{10}, K'_1, K'' encrypted using $K_{M'4}$



Group Key Update on Member Leave

ALGORITHM **1** *LEAVE*(M_I)**;**

/* THE GC CHANGES THE KEYS IN KS_I AND */

/* BROADCAST THEM TO THE GROUP MEMBERS */

$K_{ml \ldots m1} = K'_{ml \ldots m1}$; /* select a new key */

GC_broadcast($E_{K_{ml \ldots \overline{m0}}}(K_{m_l \ldots m_1})$); /* send it to M_i neighbour*/

for j = 2 **to** *l* **do**

$K_{ml \ldots mj} = K'_{ml \ldots mj}$ /* select a new key */

GC_broadcast($E_{K_{ml \ldots m_{(j+1)}}}(K_{m_l \ldots mj})$); /* broadcast to half sub-tree */

GC_broadcast($E_{K_{ml \ldots \overline{m_{(j+1)}}}}(K_{m_l \ldots mj})$); /* broadcast to other half sub-tree */

end-for;

$K = K'$; /* select a new group key */

GC_broadcast($E_{K0}(K)$); /* broadcast to 1st half tree */

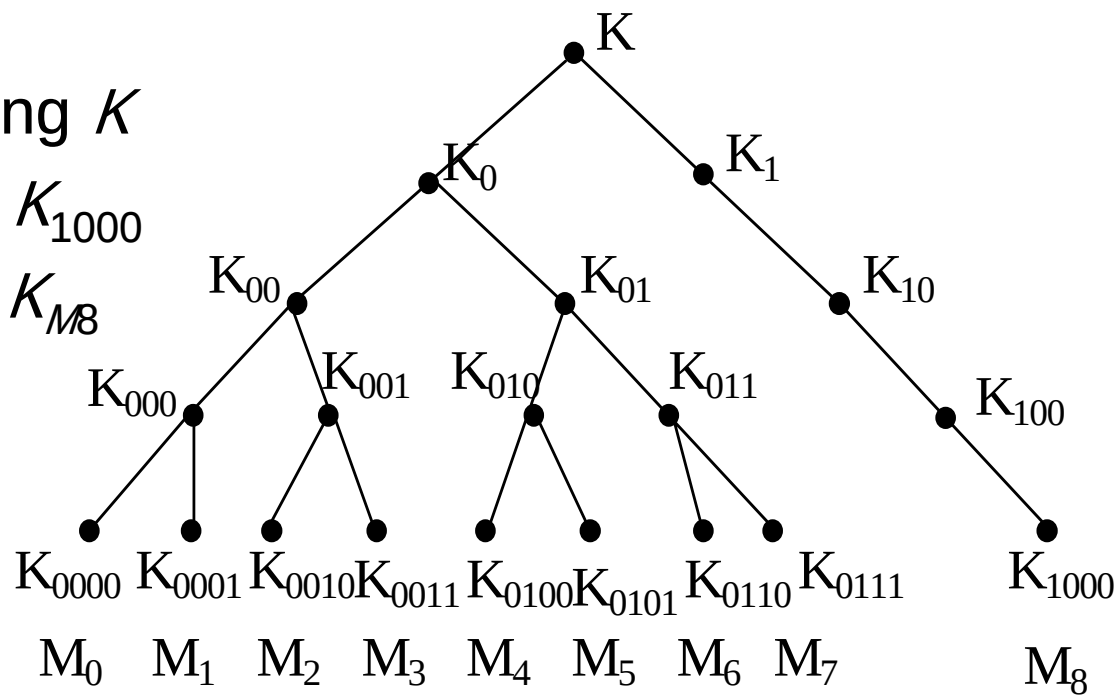
GC_broadcast($E_{K1}(K)$) ; /* broadcast to 2nd half tree */

Group size increase

- M'_8 joins:

  GC sends:

  - K encrypted using K
  - K, K_1, K_{10}, K_{100}, K_{1000} encrypted using K_{M8}



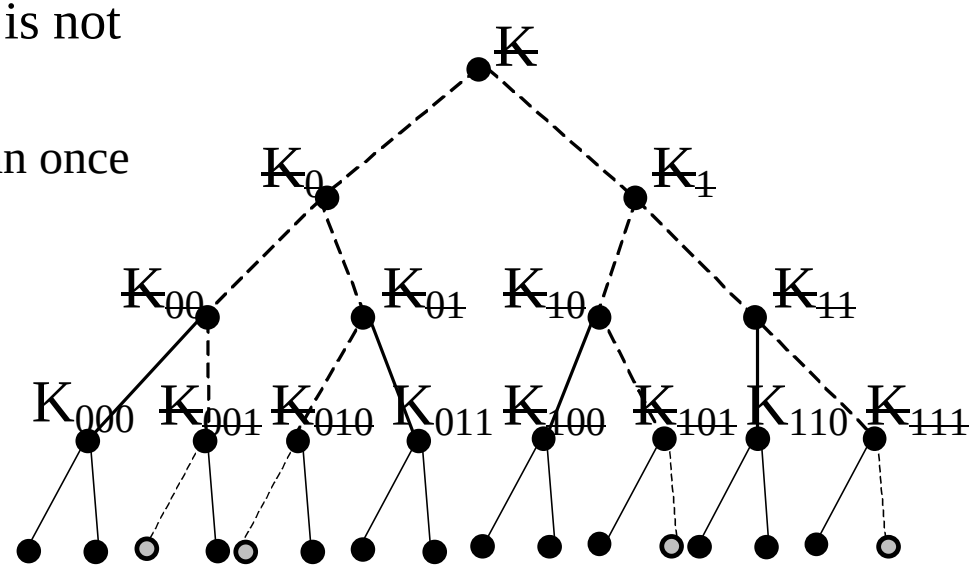
Highly Dynamic Groups

- Communication cost increases linearly with requests rate
  - Beginning and end of session
  - Sessions with short participation duration of members
  - Mobile environments where group structure is defined by the location
  - Capture of a subgroup of members
- Solution:
  - There exists an optimal tree structure that minimizes the communication complexity
  - Simultaneous processing of requests reduces communication cost because common keys need to be changed only once
  - Adaptivity:
    - Parameters: tree-structure, delay for requests processing, communication bandwidth, session type (e.g., membership dynamic, smoothness), requests history
    1. Predict probability of requests and adapt tree-structure
    2. Adapt update delay to respect bandwidth constraints

Adaptive Group Key Distribution System Highly Dynamic Groups (I)

- If multiple join/leave requests are received simultaneously, a fixed degree tree is not optimal
  - All top layers of the tree would have to be changed

- Processing requests one at a time is not optimal
  - Some keys are changed more than once



- Our approach to multiple join/leave requests:
  - Non-regular tree structure
  - Adaptive delay
  - Group dynamic prediction
- Assumption: all users have the same update probability

Optimal Tree Structure

Theorem: For a group of size 2^N the optimal tree structure has degrees: 2^k 4 4 ... 4 [4|2]. *k* depends on the probability of key update requests.

Lemma 1: If a tree $T(a_1, a_2, ..., a_t)$ is optimal then for any $1 \le i, j \le t$ the subtree $T'(a_i, ..., a_j)$ is also optimal.

Lemma 2: $T(2^{k-j}, 2^j)$ is not an optimal tree if $j > 2$.

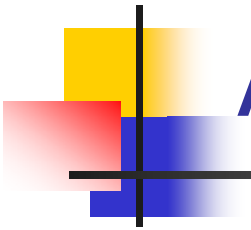
Lemma 3: T(2, 2) and T(2, 4) are not optimal.

- The optimal tree structure can be computed analytically.

Theorem: The optimal tree structure is either $2^k\ 2^2\ 2^2 ... 2^2 2^{2|1}$ or $2^{k-1}\ 2^2\ 2^2 ... 2^2 2^{2|1}$, where

$$1-(1-p)^{2^{N-(k+1)}} < p_{thresh} \le 1-(1-p)^{2^{N-k}}$$

- Results verified analytically and through simulation



Adaptive Delayed Key Update

- Tree structure change only achieves a limited bandwidth gain
- Delaying the processing of the request allows to group the processing of the requests and reduce the number of messages
- Static batch processing: Yang et al. 2001
- Adaptive delayed processing:
  - Bandwidth constraint
  - Delay constraint
- Goal:
  - Minimize bandwidth and satisfy delay constraint

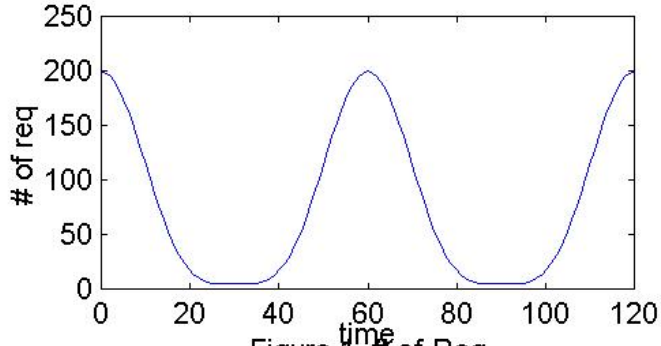


Figure 1. # of Req

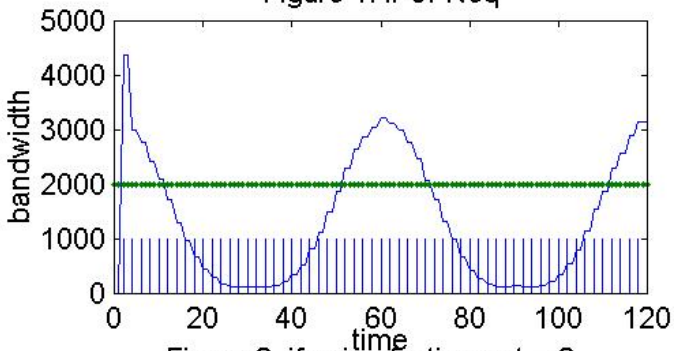


Figure 2. if using fix timeout = 2

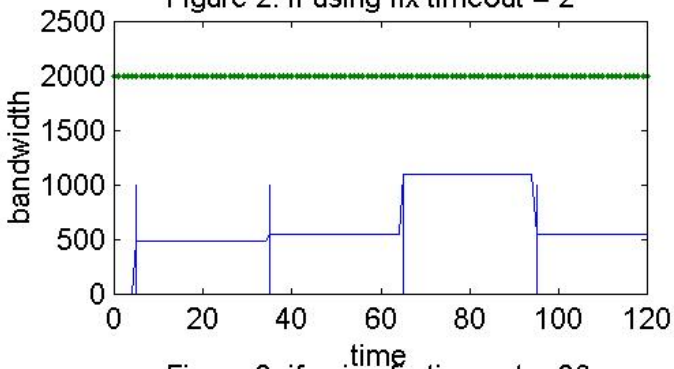


Figure 3. if using fix timeout = 30

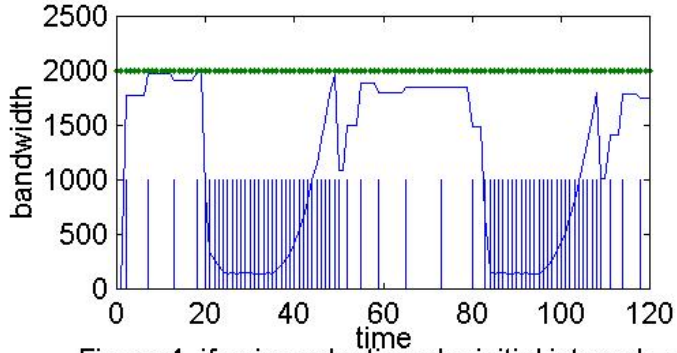


Figure 4. if using adaptive alg, initial interval = 2

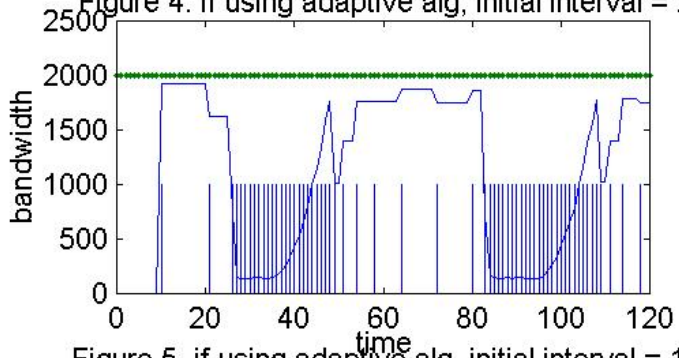


Figure 5. if using adaptive alg, initial interval = 10

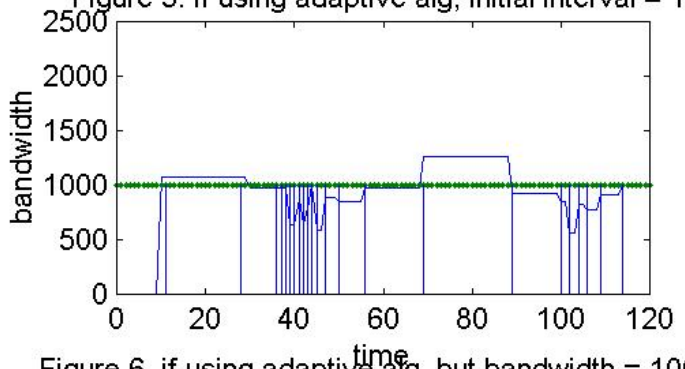
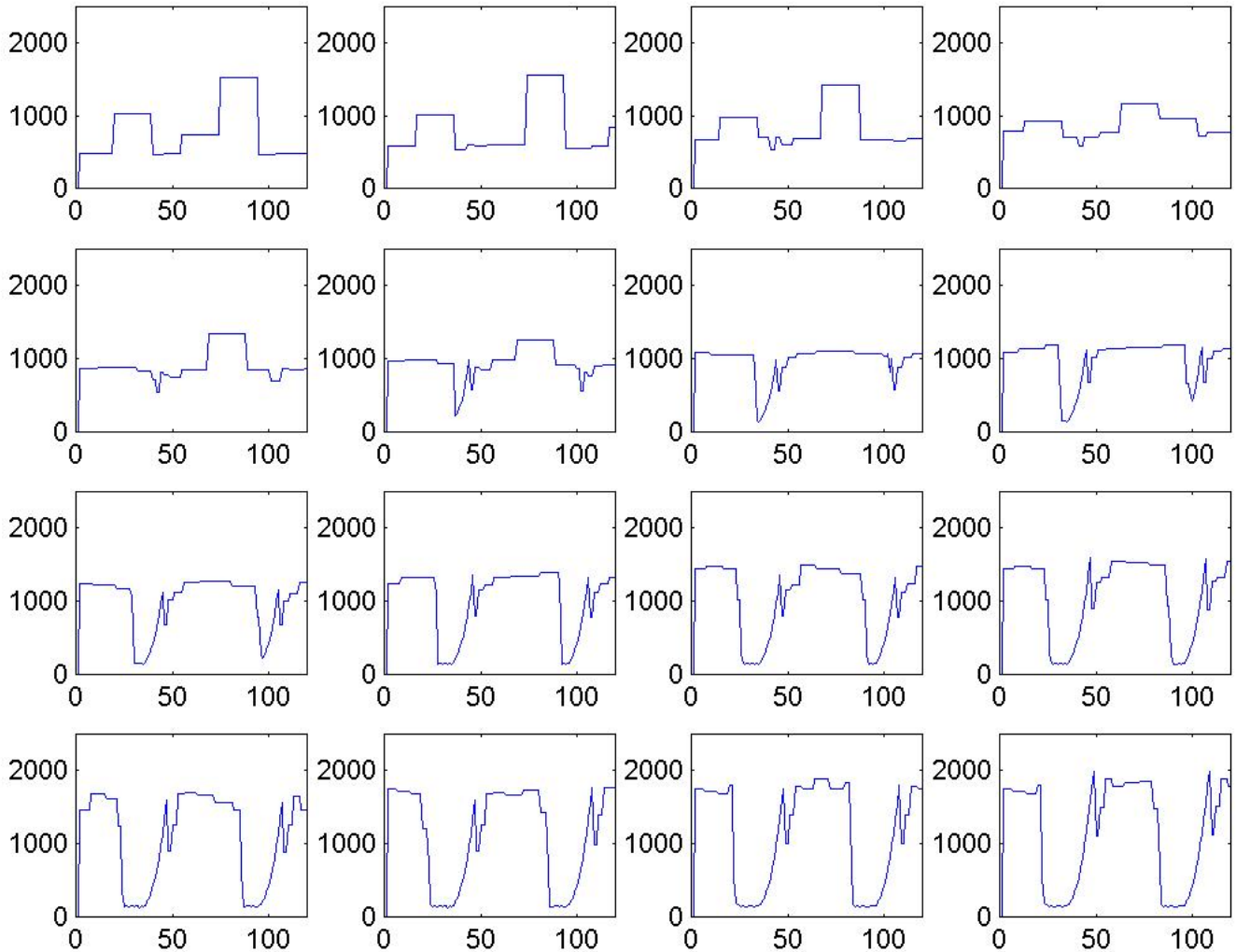


Figure 6. if using adaptive alg, but bandwidth = 1000


2000
1000
0
0
50
100