

Distributed Mutual Exclusion

- The basic requirements for mutual exclusion concerning some resource
 - At most one process may execute in the critical section at one time (safety)
 - A process requesting entry to the critical section is eventually granted it (liveness)
 - Entry to the critical section should be granted in *happened-before* order (ordering)
- The second requirement implies that deadlock and starvation do not occur

4/30/01

ICSS730 - Coordination and Agreement

1

CS Protocol

- The general protocol for entering a critical section is as follows:
 - enter()
 - Enter critical section, block if necessary
 - process()
 - Perform work
 - exit()
 - Leave critical section, other processes may now enter

4/30/01

ICSS730 - Coordination and Agreement

2

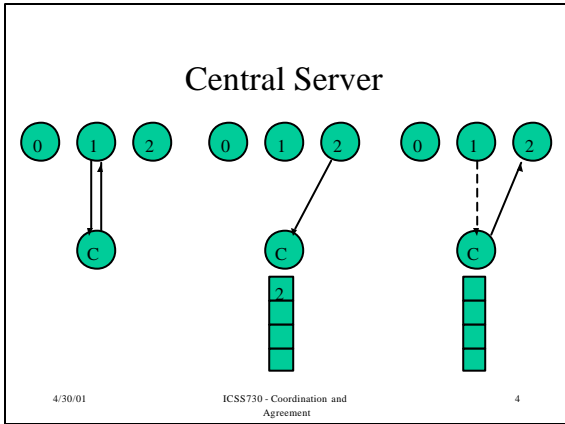
Evaluation

- Algorithm performance is measured using the following criteria
 - Bandwidth consumed
 - Client delay (at *enter* and *exit* operations)
 - Effect on the throughput of the system
 - The rate at which processes as a whole can access the critical section

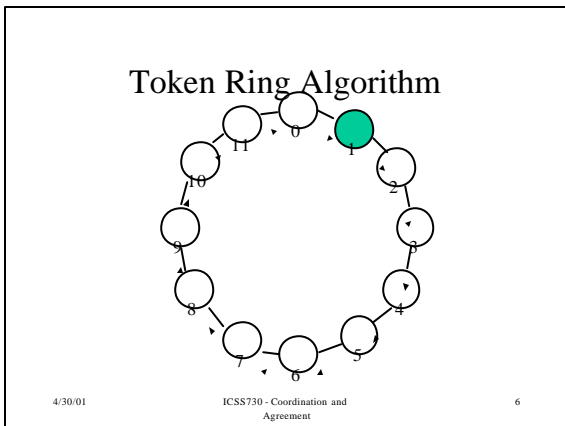
4/30/01

ICSS730 - Coordination and Agreement

3



- ### Central Server Analysis
- Meets
 - Safety, Liveness
 - Can meet ordering
 - Concerns
 - Central point of failure
 - Server might become a bottleneck
 - Failure of a client who has the token
 - Performance
 - Enter always requires two messages to be sent
 - Exit requires one release message
- 4/30/01 ICSS730 - Coordination and Agreement 5



Token Ring Analysis

- Meets
 - Safety, Liveness
- Problems
 - Loss of token
 - Process failure
- Performance
 - Constant use of network bandwidth
 - Delay to enter ranges from 0 to N

4/30/01

ICSS730 - Coordination and Agreement

7

Multicast and Logical Clocks

- Basic Idea
 - Multicast a request to enter message
 - Enter only when all processes say it is okay
- State
 - Each process has a unique identifier
 - Each process maintains a Lamport clock
- Request Format
 - $\langle T, p_i \rangle$ where T == timestamp, p_i is process id

4/30/01

ICSS730 - Coordination and Agreement

8

Ricart and Agrawals Algorithm

- Initialization
 - State := RELEASED
- Enter
 - State := WANTED
 - Multicast request to all processes
 - T := request's timestamp
 - Wait until replies received == $(n - 1)$
 - State := HELD

4/30/01

ICSS730 - Coordination and Agreement

9

Ricart and Agrawals Algorithm

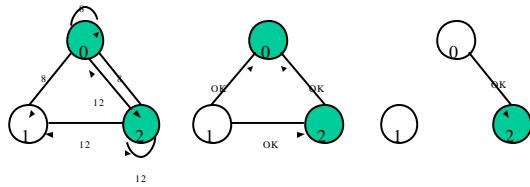
- Request $\{T, p_i\}$ received by p_j ($i < j$)
 - If State = HELD or State = WANTED and $\{T, p_j\} < \{T, p_i\}$
 - Queue request from p_i without replying
 - Else
 - Reply immediately to p_i
- Exit
 - State := RELEASED
 - Reply to any queued requests

4/30/01

ICSS730 - Coordination and Agreement

10

Algorithm In Action



4/30/01

ICSS730 - Coordination and Agreement

11

Multicast Analysis

- Meets
 - Safety, Liveness, Ordering
- Concerns
 - Single point of failure has been replaced by N
 - Obtaining the token requires $2(N-1)$ messages
 - A bottleneck can be formed by any process
 - Slower, more complicated, more expensive, and less robust
 - Like eating spinach and learning Latin in high school, some things are said to be good for you in some abstract way
 - Andrew Tannenbaum

4/30/01

ICSS730 - Coordination and Agreement

12

Maekawa's Voting Algorithm

- Processes obtain permission to enter from subsets of their peers
 - Associate with each p_i a voting set V_i such that
 - p_i is a member of V_i
 - There is at least one common member of any two voting sets
 - Each voting set has the same number of members
 - Each process is contained in M of the voting sets

4/30/01

ICSS730 - Coordination and Agreement

13

Maekawa's Algorithm

- Initialization
 - State := RELEASED
 - Voted := FALSE;
- Enter
 - State := WANTED
 - Multicast *request* to all processes in V_i
 - Wait until replies received == ($K - 1$)
 - State := HELD

4/30/01

ICSS730 - Coordination and Agreement

14

Maekawa's Algorithm

- On receipt of a *request* from p_i at p_j ($i < j$)
 - If State = HELD or Voted = TRUE
 - Queue request from p_i without replying
 - Else
 - Reply immediately to p_i
 - Voted := TRUE;

4/30/01

ICSS730 - Coordination and Agreement

15

Maekawa's Algorithm

- For p_i to exit the critical section
 - State := RELEASED
 - Multicast *release* to all processes in $V_i - \{p_i\}$
- On receipt of a *release* from p_i at p_j ($i < \rightarrow j$)
 - If queue of requests is not empty
 - Remove head of queue
 - Send reply
 - Voted := TRUE;
 - Else
 - Voted := FALSE;

4/30/01 ICSS730 - Coordination and Agreement 16

Maekawa Analysis

- Meets
 - Safety
 - Is deadlock prone
 - No ordering

4/30/01 ICSS730 - Coordination and Agreement 17

Comparison

Algorithm	Messages per exit/entry	Delay before entry (message times)	Problems
Centralized	3	2	Coordinator Crash
Distributed	$2(n-1)$	$2(n-1)$	Crash of any process
Token Ring	1 to ...	0 to $n-1$	Loss of token, process crash

4/30/01 ICSS730 - Coordination and Agreement 18

Election Algorithms

- Many distributed algorithms require one process to act as a coordinator
 - How is this process selected?
- Assumptions
 - Each process has a unique identifier
 - Every process knows the identifiers of every other process
- Election algorithms attempt to locate the process with the highest identifier and designate it as coordinator

4/30/01

ICSS730 - Coordination and Agreement

19

The Bully Algorithm

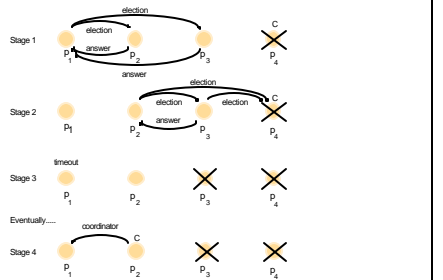
- The biggest process always wins...
- Three types of messages
 - ELECTION is sent to announce an election
 - ANSWER is sent in response to election message
 - COORDINATOR is sent to announce the winner
- The algorithm
 - P sends an ELECTION message to all processes with higher numbers
 - If no one responds, P wins and becomes the coordinator
 - If some higher numbered process replies, P is done.

4/30/01

ICSS730 - Coordination and Agreement

20

Bully in Action



4/30/01

ICSS730 - Coordination and Agreement

21

Ring Algorithm

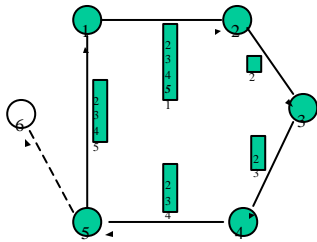
- Based on the use of a ring without a token
 - A process sends out an election message to its successor
 - Each process adds its number to the election message and sends it along
 - When the message comes back to the source, the highest numbered process in the list becomes the coordinator
 - A coordinator message is circulated to inform everyone else of the winner

4/30/01

ICSS730 - Coordination and Agreement

22

Ring in Action

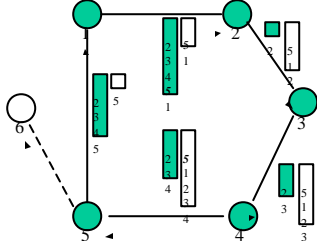


4/30/01

ICSS730 - Coordination and Agreement

23

Ring in Action

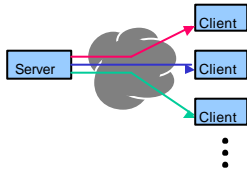


4/30/01

ICSS730 - Coordination and Agreement

24

Conventional Reliable Transport

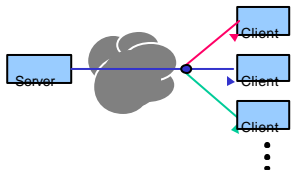


4/30/01

ICSS730 - Coordination and Agreement

25

Multicast

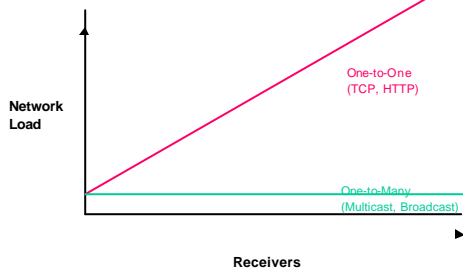


4/30/01

ICSS730 - Coordination and Agreement

26

Multicast Scales Well



4/30/01

ICSS730 - Coordination and Agreement

27

Fixes Things

- Multicast solves many problems
 - Bandwidth crisis
 - Timely Delivery
 - Latency Control
- Most applications need reliability
 - Or at least *partial* reliability

4/30/01

ICSS730 - Coordination and Agreement

28

IP Multicasting

- There are three *kinds* of IP addresses
 - Unicast
 - Broadcast
 - Multicast
- A unicast address specifies a single interface
- A broadcast address specifies all interfaces
- A multicast address specifies some of the interfaces

4/30/01

ICSS730 - Coordination and Agreement

29

The Required Pieces

- Three pieces are required for a multicast system
 - A multicast addressing scheme
 - A notification and delivery system
 - An inter-network forwarding facility

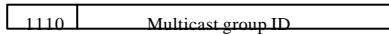
4/30/01

ICSS730 - Coordination and Agreement

30

IP Multicasting

- IP Multicasting provides two services for an application
 - Delivery to multiple destinations
 - Solicitation of servers by clients
- Class D IP addresses are used for multicast



4/30/01

ICSS730 - Coordination and Agreement

31

Host Group

- The set of hosts listening to a particular IP multicast address is called a *host group*
- A host group can span multiple networks
- Membership in the host group is dynamic
 - Hosts may join and leave at will
- No restriction on the number of hosts in a group
- A host can simply listen in on a group

4/30/01

ICSS730 - Coordination and Agreement

32

Multicast on a LAN

- Ethernet supports multicasting
 - The first byte of an Ethernet multicast address is 01
- LAN cards come in two varieties
 - Multicast filtering is done based on the hash value of the multicast hardware address
 - The card contains room to store a small, fixed, number of multicast addresses to listen for

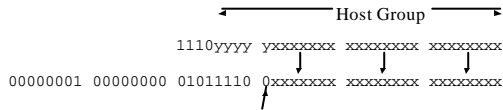
4/30/01

ICSS730 - Coordination and Agreement

33

MAC to Multicast

- IANA owns the Ethernet block
 - 00:00:5e:xx:xx:xx
- The addresses 01:00:5e:xx:xx:xx are used for multicast



Only half the block is allocated for multicast

4/30/01

ICSS730 - Coordination and Agreement

34

Example

- IP multicast address 224.0.0.2 becomes
 - 11100000.00000000.00000000.00000010
 - e0.00.00.02
 - 00.7f.ff.ff
 - 01.00.5e.00.00.02
- IP multicast address 225.0.0.2 becomes
 - 11100001.00000000.00000000.00000010
 - e1.00.00.02
 - 00.7f.ff.ff
 - 01.00.5e.00.00.02

4/30/01

ICSS730 - Coordination and Agreement

35

Beyond a Single Network

- Clearly the IP to MAC scheme only works for a single physical network
- How is the mapping done when machines from different networks are part of a host group
- The IGMP protocol is used provide multicasting between networks

4/30/01

ICSS730 - Coordination and Agreement

36

IGMP

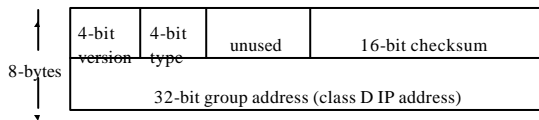
- Internet Group Management Protocol (IGMP)
 - Defined in RFC1112/RFC2236
 - Considered to be part of the IP layer
 - Messages sent in IP datagrams
 - Has a fixed-size message with no optional data

4/30/01

ICSS730 - Coordination and Agreement

37

IGMP Message



- The Current IGMP Version is 2
- IGMP Type
 - 1 is a query sent by a multicast router
 - 2 is a response sent by a host

4/30/01

ICSS730 - Coordination and Agreement

38

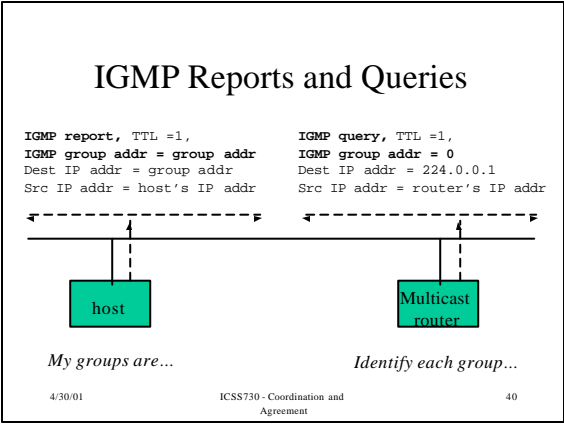
IGMP Rules

- Basic rules
 1. A host sends an IGMP report when a process first joins a group
 2. A host does not send a report when processes leave a group (even when the last process leaves a group)
 3. A multicast router sends an IGMP query at regular intervals to see if any hosts have processes belonging to any groups
 4. A host responds to a query by sending one IGMP report for each group that still has members

4/30/01

ICSS730 - Coordination and Agreement

39

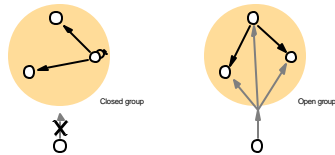


- ## Implementation Details
- There are several ways that IGMP minimizes its effect on the network
 - All communication between hosts/routers use multicast
 - A single query to request group information is sent to all groups (default rate is 125 seconds)
 - If multiple routers are on the same network, one is selected to poll membership
 - Hosts do not respond to the router's IGMP query at the same time
 - Hosts listen for responses from other hosts in the group, and suppresses unnecessary response traffic
- 4/30/01
ICSS730 - Coordination and Agreement
41

- ## Issues
- **Guaranteed Delivery**
 - Will all members of the group receive a message or will some see it and some will not?
 - **Ordering**
 - Will all members of a group see the messages delivered in the same order they were sent?
 - These are non-trivial problems
- 4/30/01
ICSS730 - Coordination and Agreement
42

System Model

- Processes are members of various groups
- Can communicate reliably over one-to-one channels



4/30/01

ICSS730 - Coordination and Agreement

43

Terminology

- Multicasting is centered on groups
 - Single/Multiple Senders
- Dynamic Group formation/management
 - Joins
 - Late Joins
 - Leaves
- Error Recovery
 - Full/Partial Repair
 - No Repair

4/30/01

ICSS730 - Coordination and Agreement

44

Basic Multicast

- Multicast(group, message)
 - For each process, p_i , in group
 - Reliably send message to p_i
- Could use threads to do this
- Ack implosion!!

4/30/01

ICSS730 - Coordination and Agreement

45

Reliable Multicast

- Satisfies the following properties
 - Integrity
 - A message is delivered at most once
 - Validity
 - A multicast message will eventually be delivered
 - Agreement
 - The message will eventually be delivered to all members of a group

4/30/01

ICSS730 - Coordination and Agreement

46

Bulletin Board Program

- Every user runs a bulletin-board application
- Every topic of discussion is a multicast group
- To post a message, the message is multicasted to the appropriate group
- Reliable multicast is required if every user is to receive every posting (eventually)

4/30/01

ICSS730 - Coordination and Agreement

47

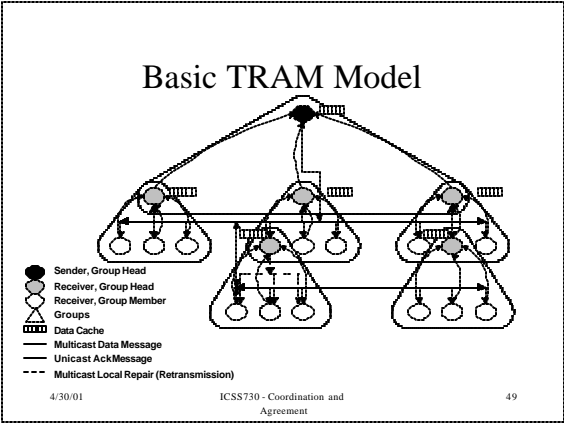
TRAM

- A tree-based reliable multicast protocol
 - Sender and receivers dynamically form repair groups
 - Repair groups are linked together to form a tree
- TRAM has been kept as lightweight as possible

4/30/01

ICSS730 - Coordination and Agreement

48



- ### Automatic Tree Formation
- The tree
 - Each receiver is associated with a repair head
 - Be able to add new receivers to the tree at any time
 - Recover from head failure through re-affiliation
 - What is a suitable repair head?
 - Shortest TTL distance
 - Eagerness to be head
 - Head experience
 - Repair data availability
- 4/30/01 ICSS730 - Coordination and Agreement 50

- ### TRAM Features
- Reliable
 - Avoids ACK implosion
 - Local Repair
 - Rate based flow control and congestion avoidance
 - Feedback to sender
 - Scalable
- 4/30/01 ICSS730 - Coordination and Agreement 51

LRMP

- The Light-Weight Reliable Multicast Protocol
 - Guarantees sequenced and reliable delivery
 - Places no restrictions on receiver's membership
 - Allows multiple senders
 - Light-weight in terms of protocol overhead and simple in control mechanisms

4/30/01

ICSS730 - Coordination and Agreement

52

Random Expanding Probe

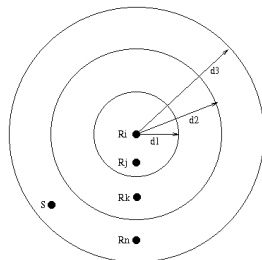
- Would prefer the repair information be as close to the receiver as possible
- REP consists of three steps
 - Divide a multicast session into hierarchical subgroups
 - Report errors to a subgroup
 - Send repairs to a subgroup

4/30/01

ICSS730 - Coordination and Agreement

53

Hierarchy of Subgroups



4/30/01

ICSS730 - Coordination and Agreement

54

LRMP

- Normal Operation
 - A source multicasts a set of data packets
 - Transmission is controlled by a transmission interval
 - A receiver detects packet loss using sequence numbers
- LRMP makes no effort to handle full repairs for late joining members

4/30/01

ICSS730 - Coordination and Agreement

55

Error Reporting in LRMP

1. Set the number of NACK request $N = 0$ and the domain level $i = 1$
2. Schedule a random timer and wait.
3. When the timer expires check
 1. If the lost packets have been received, repair terminates
 2. Otherwise if no NACK was received, send a NACK to the domain D_i
4. If D_i is not the highest level, then $i=i+1$; otherwise $N=N+1$
5. If $N < \text{Max}$, go to step 2

4/30/01

ICSS730 - Coordination and Agreement

56

LRMP Features

- Suitable for bulk data transfer
- Provides support for multiple senders
- Congestion control
- Distributed Control

4/30/01

ICSS730 - Coordination and Agreement

57

JRMS

- The Java Reliable Multicast Service
 - Enables building applications that multicast data from “senders” to “receivers” over “channels”
- Organized as a set of libraries and services for building multicast applications
- Functional components:
 - A common API which supports multiple concurrent reliable multicast transport protocols
 - Services for multicast address allocation and channel management

4/30/01

ICSS730 - Coordination and Agreement

58

Ordered Multicast

- Common ordering requirements
 - FIFO
 - If a process multicasts m1 and then m2, then every process that delivers m2 will deliver m1 before it.
 - Causal
 - If m1 is *multicasted-before* m2, then every process that delivers m2 will deliver m1 before it
 - Total
 - If a process delivers m1 before it delivers m2, then any other process that delivers m2 will deliver m1 before m2

4/30/01

ICSS730 - Coordination and Agreement

59

Bulletin Board Revisited

- FIFO
 - Every posting from a given user will be received in the same order
- Causal
 - Posting from different users, but within the same *thread* are delivered in the same order every where
- Total
 - All postings from all users would be delivered in the same order every where

4/30/01

ICSS730 - Coordination and Agreement

60

Bulletin Board Revisited

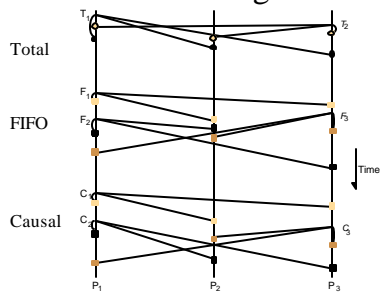
Bulletin board CS.interesting		
Item	From	Subject
23	A.Hanlon	Mach
24	G.Joseph	Microkernels
25	A.Hanlon	Re: Microkernels
26	T.L.Heureux	RPC performance
27	M.Walker	Re: Mach
end		

4/30/01

ICSS730 - Coordination and Agreement

61

Ordering



4/30/01

ICSS730 - Coordination and Agreement

62

FIFO

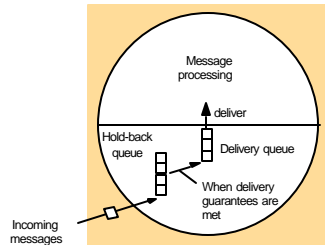
- Built on top of reliable or un-reliable multicast
- A sender assigns sequence numbers to all of its messages
- Receivers keep track of the next sequence number they expect to see
- If I get the message I expect then it is delivered, otherwise queue it

4/30/01

ICSS730 - Coordination and Agreement

63

FIFO



4/30/01

ICSS730 - Coordination and Agreement

64

Total Ordering

- Basically the same idea as FIFO except
 - Sequence numbers apply to groups instead of processes
 - Remember we are interested in ordering within a group (i.e. a group is not a newsgroup)
- How do we assign sequence numbers?

4/30/01

ICSS730 - Coordination and Agreement

65

Sequencer

```
1. Algorithm for group member p
Do initialize:  $s_g := 0$ ;
To 2D-wait/await message m in group g
  Do wait for  $g := \{ \text{requester}(p), \dots, p \}$ ;
Do B-deliver(m, i) with  $g = \text{group}(m)$ 
  Place  $\langle m, i \rangle$  in hold-back queue;
Do B-deliver("order", i, S) with  $g = \text{group}(m)$ 
  wait until  $\langle m, i \rangle$  in hold-back queue and  $S = s_g + 1$ ;
  2D-deliver m, i (after deleting it from the hold-back queue)
   $s_g := S$ ;

2. Algorithm for sequencer of g
Do initialize:  $s_g := 0$ ;
Do B-deliver(m, i) with  $g = \text{group}(m)$ 
  Do wait for  $g := \{ \text{order}, \dots, p \}$ ;
   $s_g := s_g + 1$ ;
```

4/30/01

ICSS730 - Coordination and Agreement

66

Total Ordering

The diagram illustrates the Total Ordering protocol with four processes: P1, P2, P3, and P4.

1. P1 sends a '1 Message' to P2, P3, and P4.

2. P2, P3, and P4 each send a '2 Proposed Seq' back to P1.

3. P1 collects these proposals and sends a '3 Agreed Seq' back to P2, P3, and P4.

The processes are represented by circles, and the messages are numbered 1, 2, and 3 to show the sequence of communication.

1. Message is sent with a sequence/timestamp
2. Every receiver responds with a sequence/timestamp larger than any one it has sent or received
3. Receiver collects responds and sends a commit using the largest sequence/timestamp to determine the ordering

4/30/01 ICSS730 - Coordination and Agreement 67

ISIS

- Toolkit for developing distributed applications
 - Coordinating stock trading
- Basically middleware that provides group communication primitives
- Widely quoted in the literature and used for numerous real world applications
 - Phased out in 1998

4/30/01 ICSS730 - Coordination and Agreement 68

ISIS Communication Primitives

- ABCAST
 - Total ordering using the protocol previously described
- CBCAST
 - Ordered delivery for causally related messages
- MCAST (??)
 - No ordering

4/30/01 ICSS730 - Coordination and Agreement 69

CBCAST

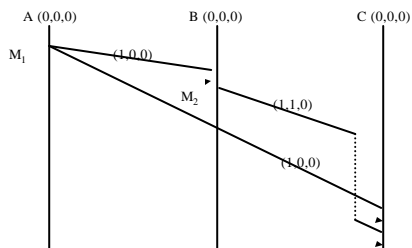
- Each process maintains a vector with one slot for each member of the group
 - The values are the sequence number of the last message number received from that process
- To send
 - Increment my slot in the vector
 - Send my vector with the message

4/30/01

ICSS730 - Coordination and Agreement

70

CBCAST



4/30/01

ICSS730 - Coordination and Agreement

71

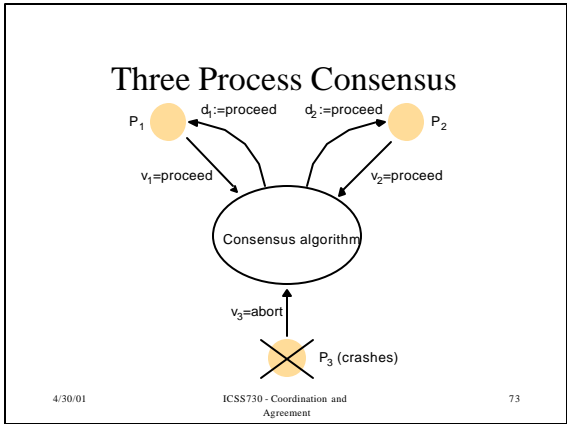
Consensus

- How do process agree on a value after one or more of the processes has proposed what the value should be?
 - Space shuttle, 3 computers, 2 say go, 1 says abort, what do you do?
- Typical system model
 - Must work even if faults occur

4/30/01

ICSS730 - Coordination and Agreement

72



- ## Requirements
- **Termination**
 - Eventually each process sets its decision variable
 - **Agreement**
 - The decision variable of all correct processes is the same
 - **Integrity**
 - If the correct processes all proposed the same value, then any correct process in the *decided* state has chosen that value
- 4/30/01 ICSS730 - Coordination and Agreement 74

byzantine

Main Entry: 1 Byz-an-tine
Pronunciation: 'bi-z&n-"tEn, 'bI, -"tIn; b&-'zan-', bI'
Function: adjective
Date: 1794

1 : of, relating to, or characteristic of the ancient city of Byzantium

2 : of, relating to, or having the characteristics of a style of architecture developed in the Byzantine Empire especially in the 5th and 6th centuries featuring the dome carried on pendentives over a square and incrustation with marble veneering and with colored mosaics on grounds of gold

3 : of or relating to the churches using a traditional Greek rite and subject to Eastern canon law

4 often not capitalized a : of, relating to, or characterized by a devious and usually surreptitious manner of operation <a Byzantine power struggle> b : intricately involved : LABYRINTHINE <rules of Byzantine complexity>

4/30/01 ICSS730 - Coordination and Agreement 75

Byzantine Generals

- Three or more commanders agree to attack or retreat
- One, the commander, issues the order.
- The others are to agree to attack or retreat
 - But one or more of the generals is treacherous in they tell one general to attack and the other to retreat
- Differs in that one process proposes a value that the others are to agree on. As opposed to each proposing a value.

4/30/01

ICSS730 - Coordination and Agreement

76

Requirements

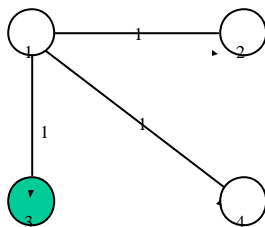
- Termination
 - Eventually each correct process sets its decision variable
- Agreement
 - The decision value of all correct processes is the same
- Integrity
 - If the commander is correct, then all correct processes decide on the value proposed by the commander

4/30/01

ICSS730 - Coordination and Agreement

77

Lamport Solution

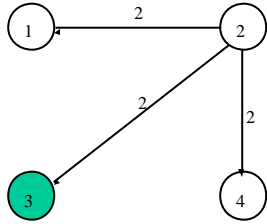


4/30/01

ICSS730 - Coordination and Agreement

78

Lamport Solution

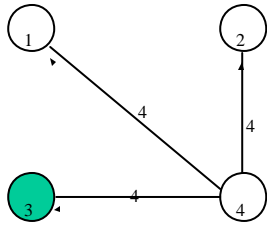


4/30/01

ICSS730 - Coordination and Agreement

79

Lamport Solution

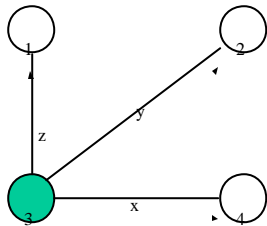


4/30/01

ICSS730 - Coordination and Agreement

80

Lamport Solution



4/30/01

ICSS730 - Coordination and Agreement

81

Vectors

1	Got (1,2,z,4)
2	Got (1,2,y,4)
3	Got (1,2,3,4)
4	Got (1,2,x,4)

4/30/01

ICSS730 - Coordination and Agreement

82

Consolidate

1	2	4
(1,2,z,4)	(1,2,z,4)	(1,2,z,4)
(1,2,y,4)	(1,2,y,4)	(1,2,y,4)
(a,b,c,d)	(e,f,g,h)	(i,j,k,l)
(1,2,x,4)	(1,2,x,4)	(1,2,x,4)

Result \rightarrow (1,2,UNKNOWN,4)

4/30/01

ICSS730 - Coordination and Agreement

83

Issues

- Agreement is possible only if *more* than two-thirds of the processors are working properly
- No agreement is possible in a system with asynchronous processors and unbounded transmission delays
 - Slow processors appear to be dead

4/30/01

ICSS730 - Coordination and Agreement

84
