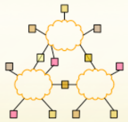
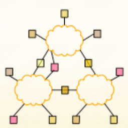
15-446 Distributed Systems Spring 2009



Final Review

15-446 Distributed Systems Spring 2009



L-14 Security

Important Lessons - Security

- Internet design and growth → security challenges
- Symmetric (pre-shared key, fast) and asymmetric (key pairs, slow) primitives provide:
  - Confidentiality
  - Integrity
  - Authentication
- “Hybrid Encryption” leverages strengths of both.
- Great complexity exists in securely acquiring keys.
- Crypto is hard to get right, so use tools from others, don’t design your own (e.g. TLS).

Two ways to cut a table (ACM)

- Order by columns (ACL) or rows (Capability Lists)?

	File1	File2	File3
Ann	*rx*	*r*	rwx
Bob	*rwxo*	r	--
Charlie	*rx*	rwo	w

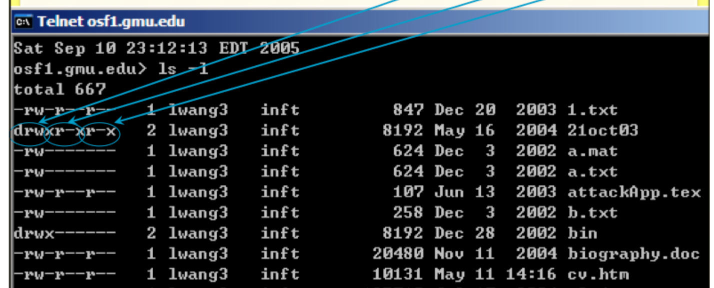
ACLs (↓ columns)

Capability (→ rows)

5

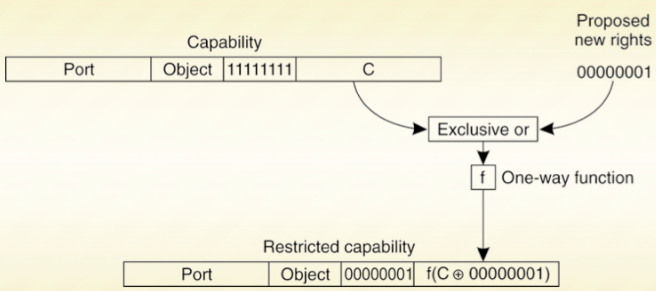
ACL:Default Permission and Abbreviation

- Example: UNIX ➔
- Three classes of users: owner, group, all others



6

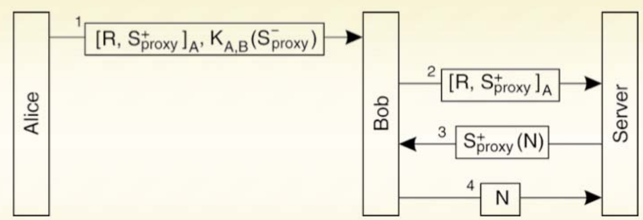
Capabilities and Attribute Certificates (2)



- Generation of a restricted capability from an owner capability.

7

Delegation (2)



- Using a proxy to delegate and prove ownership of access rights.

8

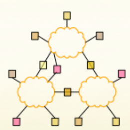
Sybil Attack undermines assumed mapping between identity to entity and hence number of faulty entities

- A Sybil attack is the forging of multiple identities for malicious intent -- having a set of faulty entities represented through a larger set of identities.
- The purpose of such an attack is to compromise a disproportionate share of a system.
- Result is overthrowing of any assumption of designed reliably based on a limited proportion of faulty entities.

9

10

15-446 Distributed Systems Spring 2009



L-15 Fault Tolerance

11

Important Lessons

- Terminology & Background
  - Failure models
- Byzantine Fault Tolerance
  - Protocol design → with and without crypto
  - How many servers do we need to tolerate
- Issues in client/server
  - Where do all those RPC failure semantics come from?
- Reliable group communication
  - How do we manage group membership changes as part of reliable multicast

12

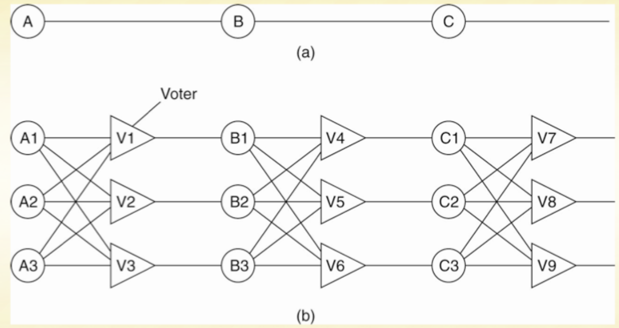
Failure Models

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure	A server fails to respond to incoming requests
Receive omission	A server fails to receive incoming messages
Send omission	A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure	A server's response is incorrect
Value failure	The value of the response is wrong
State transition failure	The server deviates from the correct flow of control
Arbitrary failure	A server may produce arbitrary responses at arbitrary times

- A system is said to fail if it cannot meet its promises. An error on the part of a system's state may lead to a failure. The cause of an error is called a fault.

13

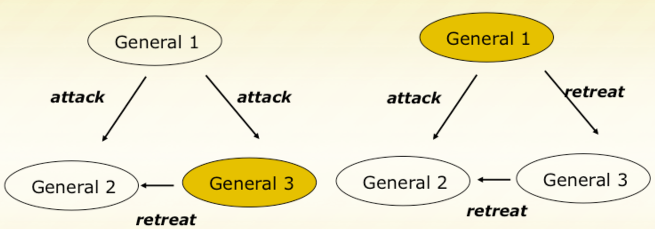
Failure Masking by Redundancy



- Triple modular redundancy. For each voter, if two or three of the inputs are the same, the output is equal to the input. If all three inputs are different, the output is undefined.

14

Impossibility Results

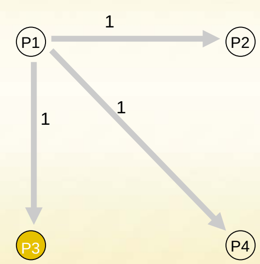


- *No solution for three processes can handle a single traitor.*
- *In a system with m faulty processes agreement can be achieved only if there are 2m+1 (more than 2/3) functioning correctly.*

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401. 15

Byzantine General Problem Example - 1

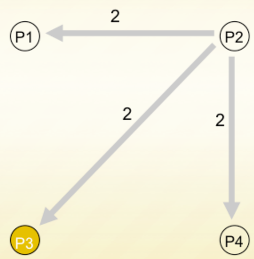
- Phase 1: Generals announce their troop strengths to each other



16

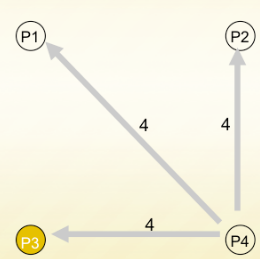
Byzantine General Problem Example - 2

- Phase 1: Generals announce their troop strengths to each other



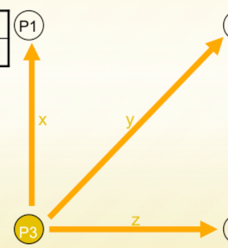
Byzantine General Problem Example - 3

- Phase 1: Generals announce their troop strengths to each other



Byzantine General Problem Example - 4

- Phase 2: Each general construct a vector with all troops



P1:

P1	P2	P3	P4
1	2	x	4

P2:

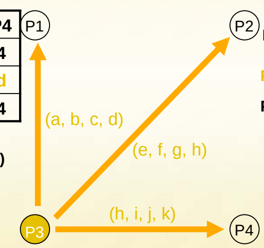
P1	P2	P3	P4
1	2	y	4

P4:

P1	P2	P3	P4
1	2	z	4

Byzantine General Problem Example - 5

- Phase 3,4: Generals send their vectors to each other and compute majority voting



P1:

	P1	P2	P3	P4
P2	1	2	y	4
P3	a	b	c	d
P4	1	2	z	4
	(1,	2,	?,	4)

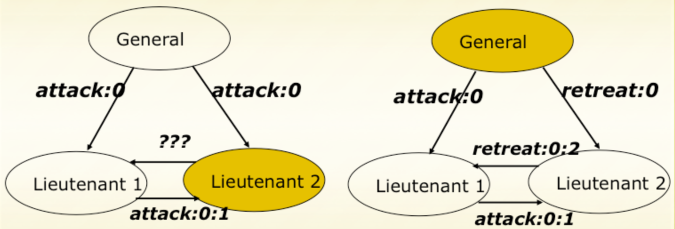
P2:

	P1	P2	P3	P4
P1	1	2	x	4
P3	e	f	g	h
P4	1	2	z	4
	(1,	2,	?,	4)

P4:

	P1	P2	P3	P4
P1	1	2	x	4
P2	1	2	y	4
P3	h	i	j	k
	(1,	2,	?,	4)

Signed messages



SM(1) with one traitor

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401. 21

Server Crashes (3)

- Consider scenario where a client sends text to a print server.
- There are three events that can happen at the server:
  - Send the completion message (M),
  - Print the text (P),
  - Crash (C) – at recovery, send 'recovery' message to clients.
- Server strategies:
  - send completion message before printing
  - send completion message after printing

22

Server Crashes (4)

- These events can occur in six different orderings:

1. M →P →C: A crash occurs after sending the completion message and printing the text.
2. M →C (→P): A crash happens after sending the completion message, but before the text could be printed.
3. P →M →C: A crash occurs after sending the completion message and printing the text.
4. P→C(→M): The text printed, after which a crash occurs before the completion message could be sent.
5. C (→P →M): A crash happens before the server could do anything.
6. C (→M →P): A crash happens before the server could do anything.

23

Server Crashes (6)

- Different combinations of client and server strategies in the presence of server crashes.

Client	Server					
	Strategy M → P			Strategy P → M		
Reissue strategy	MPC	MC(P)	C(MP)	PMC	PC(M)	C(PM)
Always	DUP	OK	OK	DUP	DUP	OK
Never	OK	ZERO	ZERO	OK	OK	ZERO
Only when ACKed	DUP	OK	ZERO	DUP	OK	ZERO
Only when not ACKed	OK	ZERO	OK	OK	DUP	OK

OK = Text is printed once
DUP = Text is printed twice
ZERO = Text is not printed at all

24

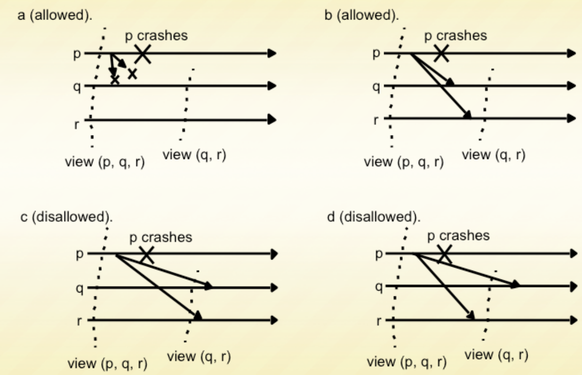
Client Crashes

- Can create orphans (unwanted computations) that waste CPU, potentially lock up resources and create confusion when client re-boots.
- Nelson solutions:
  1. Orphan Extermination – keep a log of RPCs at client that is checked at re-boot time to remove orphans.
  2. Reincarnation – divide time into epochs. After a client re-boot, increment its epoch and kill off any of its requests belonging to an earlier epoch.
  3. Gentle Reincarnation – at reboot time, an epoch announcement causes all machines to locate the owners of any remote computations.
  4. Expiration – each RPC is given time T to complete (but a live client can ask for more time)

Nelson. Remote Procedure Call. Ph.D. Thesis, CMU, 1981.

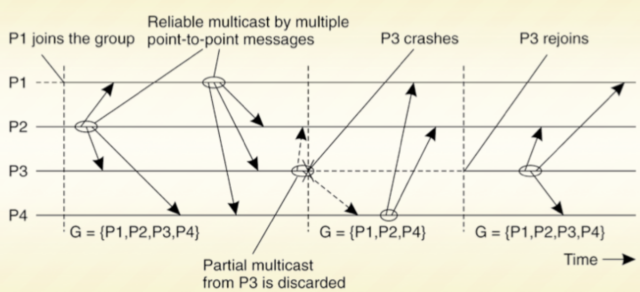
25

View-synchronous group communication



26

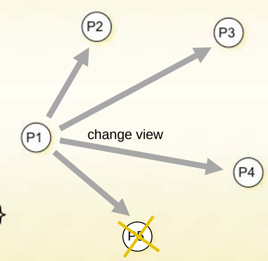
Virtual Synchrony (2)



27

Virtual Synchrony Implementation: Example

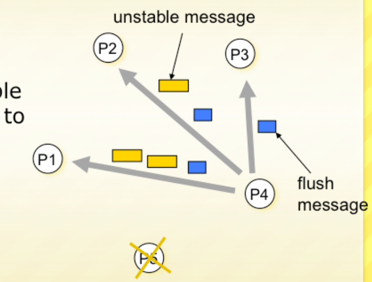
- G_i = {P1, P2, P3, P4, P5}
- P5 fails
- P1 detects that P5 has failed
- P1 send a "view change" message to every process in G_{i+1} = {P1, P2, P3, P4}



28

Virtual Synchrony Implementation: Example

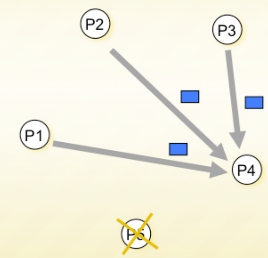
- Every process
  - Send each unstable message m from Gi to members in G_{i+1}
  - Marks m as being stable
  - Send a flush message to mark that all unstable messages have been sent



29

Virtual Synchrony Implementation: Example

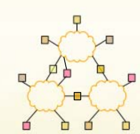
- Every process
  - After receiving a flush message from any process in G_{i+1} installs G_{i+1}



30

31

15-446 Distributed Systems Spring 2009



L-16 Transactions

32

Transactions – The ACID Properties

- Are the four desirable properties for reliable handling of concurrent transactions.
- Atomicity
  - The "All or Nothing" behavior.
- C: stands for either
  - Concurrency: Transactions can be executed concurrently
  - … or Consistency: Each transaction, if executed by itself, maintains the correctness of the database.
- Isolation (Serializability)
  - Concurrent transaction execution should be equivalent (in effect) to a *serialized* execution.
- Durability
  - Once a transaction is *done*, it stays done.

33

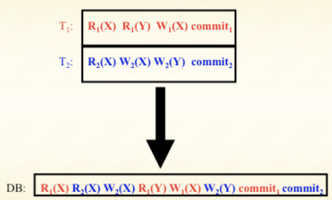
Transaction life histories

Successful	*Aborted by client*	*Aborted by server*		
openTransaction	*openTransaction*			*openTransaction*
operation	*operation*			*operation*
operation	*operation*			*operation*
:	:	server aborts transaction	→	:
operation	*operation*			*operation ERROR reported to client*
closeTransaction	*abortTransaction*			

- *openTransaction()* → *trans;*
  - starts a new transaction and delivers a unique TID *trans*. This identifier will be used in the other operations in the transaction.
- *closeTransaction(trans)* → *(commit, abort);*
  - ends a transaction: a *commit* return value indicates that the transaction has committed; an *abort* return value indicates that it has aborted.
- *abortTransaction(trans);*
  - aborts the transaction.

34

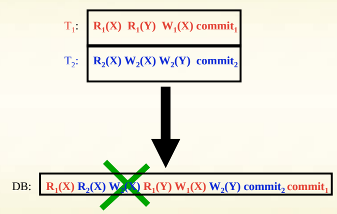
Need for serializable execution



Data manager interleaves operations to improve concurrency

35

Non serializable execution

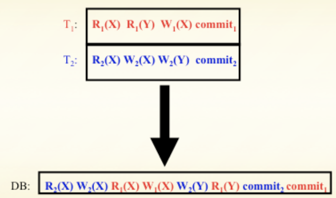


Unsafe! Not serializable

Problem: transactions may "interfere". Here, T_2 changes x, hence T_1 should have either run first (read and write) or after (reading the changed value).

36

Serializable execution



Data manager interleaves operations to improve concurrency but schedules them so that it looks as if one transaction ran at a time. This schedule "looks" like T_2 ran first.

37

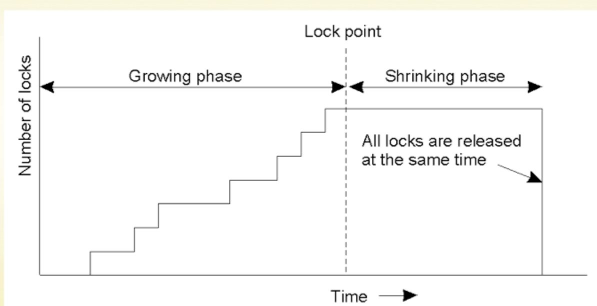
What about the locks?

- Unlike other kinds of distributed systems, transactional systems typically *lock* the data they access
- They obtain these locks as they run:
  - Before accessing "x" get a lock on "x"
  - Usually we assume that the application knows enough to get the right kind of lock. It is not good to get a read lock if you'll later need to update the object
- In clever applications, one lock will often cover many objects

38

Strict Two-Phase Locking (2)

- Strict two-phase locking.



39

Lock compatibility

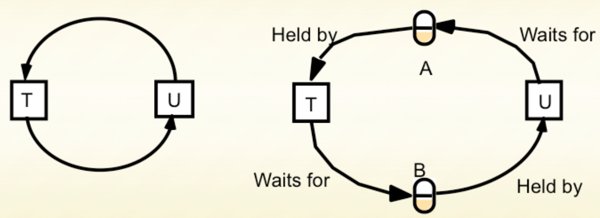
For one object		*Lock requested*	
		read	*write*
Lock already set	*none*	OK	OK
	read	OK	wait
	write	wait	wait

Operation Conflict rules:

1. If a transaction T has already performed a read operation on a particular object, then a concurrent transaction U must not write that object until T commits or aborts
2. If a transaction T has already performed a read operation on a particular object, then a concurrent transaction U must not read or write that object until T commits or aborts

40

The wait-for graph



41

Dealing with Deadlock in two-phase locking

- Deadlock prevention
  - Acquire all needed locks in a single atomic operation
  - Acquire locks in a particular order
- Deadlock detection
  - Keep graph of locks held. Check for cycles periodically or each time an edge is added
  - Cycles can be eliminated by aborting transactions
- Timeouts
  - Aborting transactions when time expires

42

Contrast: Timestamped approach

- Using a fine-grained clock, assign a "time" to each transaction, uniquely. E.g. T1 is at time 1, T2 is at time 2
- Now data manager tracks temporal history of each data item, responds to requests as if they had occured at time given by timestamp
- At commit stage, make sure that commit is consistent with serializability and, if not, abort

43

Contrast: Timestamped approach

- Using a fine-grained clock, assign a "time" to each transaction, uniquely. E.g. T1 is at time 1, T2 is at time 2
- Now data manager tracks temporal history of each data item, responds to requests as if they had occured at time given by timestamp
- At commit stage, make sure that commit is consistent with serializability and, if not, abort

44

Two Phase Commit Protocol - 6

Recovery

- 'Wait' in Coordinator – use a time-out mechanism to detect participant crashes. Send GLOBAL_ABORT
- 'Init' in Participant – Can also use a time-out and send VOTE_ABORT
- 'Ready' in Participant P – abort is not an option (since already voted to COMMIT and so coordinator might eventually send GLOBAL_COMMIT). Can contact another participant Q and choose an action based on its state.

State of Q	Action by P
COMMIT	Transition to COMMIT
ABORT	Transition to ABORT
INIT	Both P and Q transition to ABORT (Q sends VOTE_ABORT)
READY	Contact more participants. If all participants are 'READY', must wait for coordinator to recover

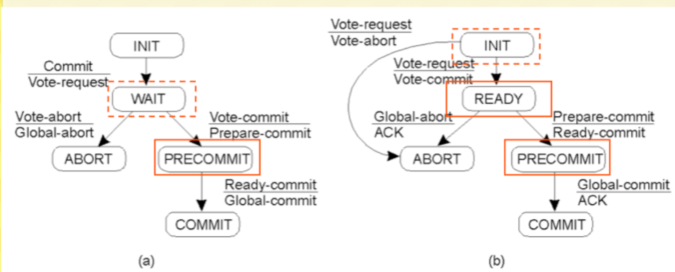
45

Three Phase Commit protocol - 1

- Problem with 2PC
  - If coordinator crashes, participants cannot reach a decision, stay blocked until coordinator recovers
- Three Phase Commit3PC
  - There is no single state from which it is possible to make a transition directly to either COMMIT or ABORT states
  - There is no state in which it is not possible to make a final decision, and from which a transition to COMMIT can be made

46

Three-Phase Commit protocol - 2



a) Finite state machine for the coordinator in 3PC
b) Finite state machine for a participant

47

Three Phase Commit Protocol - 3

Recovery

- 'Wait' in Coordinator – same
- 'Init' in Participant – same
- 'PreCommit' in Coordinator – Some participant has crashed but we know it wanted to commit. GLOBAL_COMMIT the application knowing that once the participant recovers, it will commit.
- 'Ready' or 'PreCommit' in Participant P – (i.e. P has voted to COMMIT)

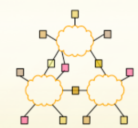
State of Q	Action by P
PRECOMMIT	Transition to PRECOMMIT. If all participants in PRECOMMIT, if majority in PRECOMMIT can COMMIT the transaction
ABORT	Transition to ABORT
INIT	Both P (in READY) and Q transition to ABORT (Q sends VOTE_ABORT)
READY	Contact more participants. If can contact a majority and they are in 'Ready', then ABORT the transaction. If the participants contacted in 'PreCommit' it is safe to COMMIT the transaction

Note: if any participant is in state PRECOMMIT, it is impossible for any other participant to be in any state other than READY or PRECOMMIT.

48

49

15-446 Distributed Systems Spring 2009



L-17 Distributed File Systems

50

Wrap up: Design Issues

- Name space
- Authentication
- Caching
- Consistency
- Locking

51

NFS V2 Design

- "Dumb", "Stateless" servers
- Smart clients
- Portable across different OSs
- Immediate commitment and idempotency of operations
- Low implementation cost
- Small number of clients
- Single administrative domain

52

Stateless File Server?

- Statelessness
  - Files are state, but…
  - Server exports files without creating extra state
    - No list of "who has this file open" (permission check on each operation on open file!)
    - No "pending transactions" across crash
- Results
  - Crash recovery is "fast"
    - Reboot, let clients figure out what happened
  - Protocol is "simple"
- State stashed elsewhere
  - Separate MOUNT protocol
  - Separate NLM locking protocol

53

NFS V2 Operations

- V2:
  - NULL, GETATTR, SETATTR
  - LOOKUP, READLINK, READ
  - CREATE, WRITE, REMOVE, RENAME
  - LINK, SYMLINK
  - READIR, MKDIR, RMDIR
  - STATFS (get file system attributes)

54

AFS Assumptions

- Client machines are un-trusted
  - Must **prove** they act for a specific user
    - Secure RPC layer
  - Anonymous "system:anyuser"
- Client machines have disks(!!)
  - Can cache whole files over long periods
- Write/write and write/read sharing are rare
  - Most files updated by one user, on one machine

55

Topic 1: Name-Space Construction and Organization

- NFS: per-client linkage
  - Server: export /root/fs1/
  - Client: mount server:/root/fs1 /fs1 → fhandle
- AFS: global name space
  - Name space is organized into Volumes
    - Global directory /afs;
    - /afs/cs.wisc.edu/vol1/…; /afs/cs.stanford.edu/vol1/…
  - Each file is identified as fid = <vol_id, vnode #, uniquifier>
  - All AFS servers keep a copy of "volume location database", which is a table of vol_id→ server_ip mappings

56

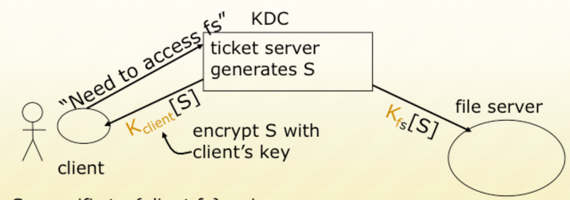
Topic 2: User Authentication and Access Control

- User X logs onto workstation A, wants to access files on server B
  - How does A tell B who X is?
  - Should B believe A?
- Choices made in NFS V2
  - All servers and all client workstations share the same <uid, gid> name space → B send X's <uid,gid> to A
    - Problem: root access on any client workstation can lead to creation of users of arbitrary <uid, gid>
  - Server believes client workstation unconditionally
    - Problem: if any client workstation is broken into, the protection of data on the server is lost;
    - <uid, gid> sent in clear-text over wire → request packets can be faked easily

57

A Better AAA System: Kerberos

- Basic idea: shared secrets
  - User proves to KDC who he is; KDC generates shared secret between client and file server



S: specific to {client,fs} pair;
"short-term session-key"; expiration time (e.g. 8 hours)

58

AFS ACLs

- Apply to directory, not to file
- Format:
  - sseshan rlidwka
  - srini@cs.cmu.edu rl
  - sseshan:friends rl
- Default realm is typically the cell name (here andrew.cmu.edu)
- Negative rights
  - Disallow "joe rl" even though joe is in sseshan:friends

59

Topic 3: Client-Side Caching

- Why is client-side caching necessary?
- What is cached
  - Read-only file data and directory data → easy
  - Data written by the client machine → when is data written to the server? What happens if the client machine goes down?
  - Data that is written by other machines → how to know that the data has changed? How to ensure data consistency?
  - Is there any pre-fetching?

60

Client Caching in NFS v2

- Cache both clean and dirty file data and file attributes
- File attributes in the client cache expire after 60 seconds (file data doesn't expire)
- File data is checked against the modified-time in file attributes (which could be a cached copy)
  - Changes made on one machine can take up to 60 seconds to be reflected on another machine
- Dirty data are buffered on the client machine until file close or up to 30 seconds
  - If the machine crashes before then, the changes are lost
  - Similar to UNIX FFS local file system behavior

61

Implication of NFS v2 Client Caching

- Data consistency guarantee is very poor
  - Simply unacceptable for some distributed applications
  - Productivity apps tend to tolerate such loose consistency
- Different client implementations implement the "prefetching" part differently
- Generally clients do not cache data on local disks

62

Client Caching in AFS v2

- Client caches both clean and dirty file data and attributes
  - The client machine uses local disks to cache data
  - When a file is opened for read, the whole file is fetched and cached on disk
    - Why? What's the disadvantage of doing so?
- However, when a client caches file data, it obtains a "callback" on the file
- In case another client writes to the file, the server "breaks" the callback
  - Similar to invalidations in distributed shared memory implementations
- Implication: file server must keep state!

63

Semantics of File Sharing

Method	Comment
UNIX semantics	Every operation on a file is instantly visible to all processes
Session semantics	No changes are visible to other processes until the file is closed
Immutable files	No updates are possible; simplifies sharing and replication
Transactions	All changes occur atomically

- Four ways of dealing with the shared files in a distributed system.

64

Session Semantics in AFS v2

- What it means:
  - A file write is visible to processes on the same box immediately, but not visible to processes on other machines until the file is closed
  - When a file is closed, changes are visible to new opens, but are not visible to "old" opens
  - All other file operations are visible everywhere immediately
- Implementation
  - Dirty data are buffered at the client machine until file close, then flushed back to server, which leads the server to send "break callback" to other clients

65

File Locking (3)

		Requested file denial state			
		NONE	READ	WRITE	BOTH
Current access state	READ	Succeed	Fail	Succeed	Fail
	WRITE	Succeed	Succeed	Fail	Fail
	BOTH	Succeed	Fail	Fail	Fail

- The result of an open operation with share reservations in NFS → When the client requests a denial state given the current file access state.

66

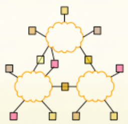
Failure recovery

- What if server fails?
  - Lock holders are expected to re-establish the locks during the "grace period", during which no other locks are granted
- What if a client holding the lock fails?
- What if network partition occurs?
- NFS relies on "network status monitor" for server monitoring

67

68

15-446 Distributed Systems Spring 2009



L-18 More DFS

Hardware Model

- CODA and AFS assume that client workstations are personal computers controlled by their user/owner
  - ***Fully autonomous***
  - ***Cannot be trusted***
- CODA allows owners of laptops to operate them in ***disconnected mode***
  - ***Opposite of ubiquitous connectivity***

70

Pessimistic Replica Control

- Would require client to acquire ***exclusive*** (RW) or ***shared*** (R) control of cached objects before accessing them in disconnected mode:
  - Acceptable solution for voluntary disconnections
  - Does not work for involuntary disconnections
- What if the laptop remains disconnected for a long time?

71

Leases

- We could grant exclusive/shared control of the cached objects for a ***limited amount of time***
- Works very well in ***connected mode***
  - Reduces server workload
  - Server can keep leases in volatile storage as long as their duration is shorter than boot time
- Would only work for very short disconnection periods

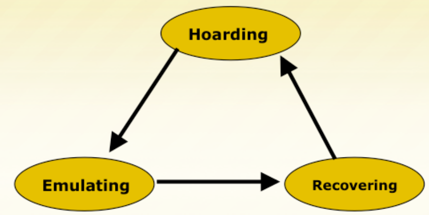
72

Optimistic Replica Control (I)

- ***Optimistic replica control*** allows access in ***every*** disconnected mode
  - Tolerates temporary inconsistencies
  - Promises to detect them later
  - Provides ***much higher data availability***

73

Coda (Venus) States



1. ***Hoarding:***
   Normal operation mode
2. ***Emulating:***
   Disconnected operation mode
3. ***Reintegrating:***
   Propagates changes and detects inconsistencies

74

Reintegration

- When workstation gets reconnected, Coda initiates a ***reintegration process***
  - Performed one volume at a time
  - Venus ships replay log to all volumes
  - Each volume performs a log replay algorithm
- Only care write/write confliction
  - Succeed?
    - Yes. Free logs, reset priority
    - No. Save logs to a tar. Ask for help

75

Performance

- Duration of Reintegration
  - A few hours disconnection → 1 min
  - But sometimes much longer
- Cache size
  - 100MB at client is enough for a "typical" workday
- Conflicts
  - No Conflict at all! Why?
  - Over 99% modification by the same person
  - Two users modify the same obj within a day: <0.75%

76

Working on slow networks

- Make local copies
  - Must worry about update conflicts
- Use remote login
  - Only for text-based applications
- Use instead a LBFS
  - Better than remote login
  - Must deal with issues like auto-saves blocking the editor for the duration of transfer

77

LBFS design

- Provides close-to-open consistency
- Uses a large, persistent file cache at client
  - Stores clients working set of files
- LBFS server divides file it stores into chunks and indexes the chunks by hash value
- Client similarly indexes its file cache
- Exploits similarities between files
  - LBFS never transfers chunks that the recipient already has

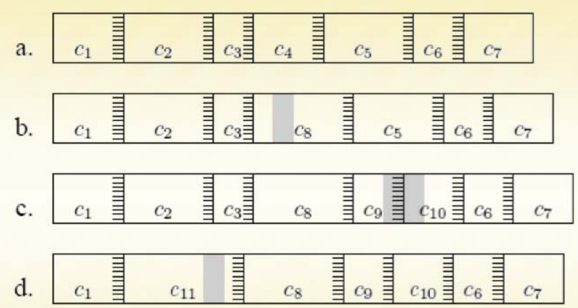
78

Indexing

- Uses the SHA-1 algorithm for hashing
  - It is collision resistant
- Central challenge in indexing file chunks is keeping the index at a reasonable size while dealing with shifting offsets
  - Indexing the hashes of fixed size data blocks
  - Indexing the hashes of all overlapping blocks at all offsets

79

Effects of edits on file chunks

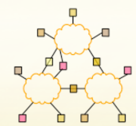


- Chunks of file before/after edits
  - Grey shading show edits
- Stripes show 48byte regions with magic hash values creating chunk boundaries

80

81

15-446 Distributed Systems Spring 2009



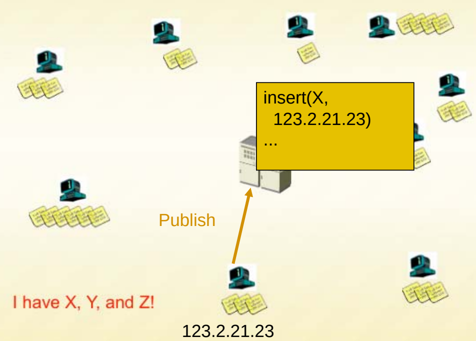
L-19 P2P

What's out there?

	Central	Flood	Super-node flood	Route
Whole File	Napster	Gnutella		Freenet
Chunk Based	BitTorrent		KaZaA (bytes, not chunks)	DHTs

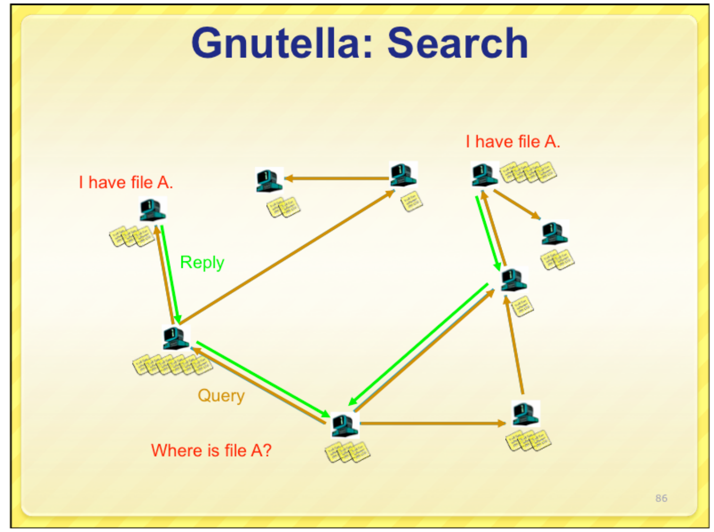
83

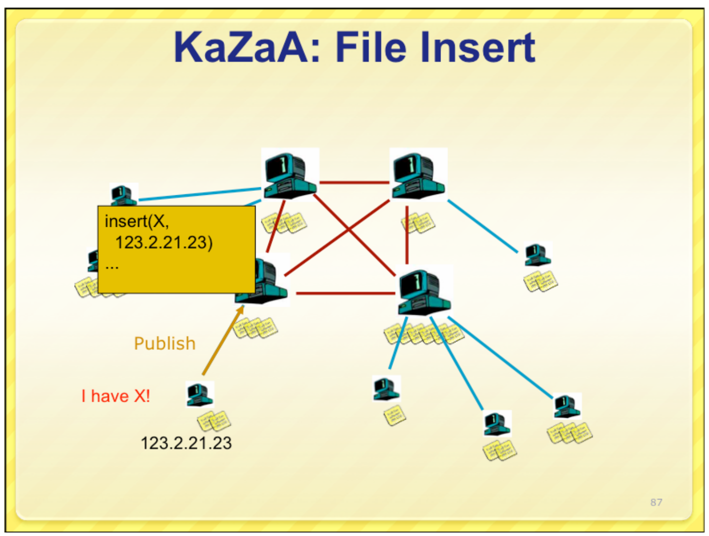
Napster: Publish

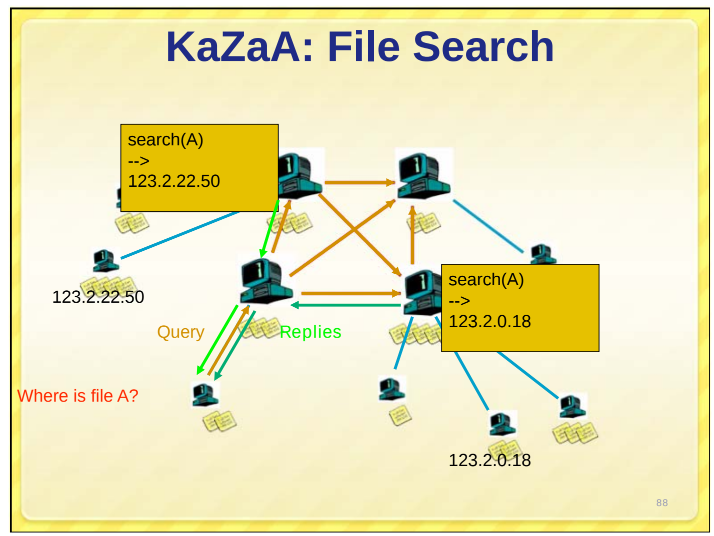


84

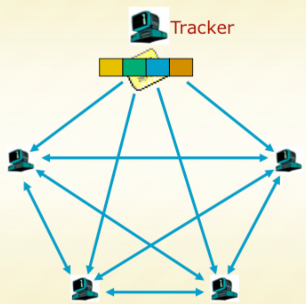

Napster: Search
123.2.0.18
search(A)
-->
123.2.0.18
Fetch
Query
Reply
Where is file A?
85


Gnutella: Search
I have file A.
I have file A.
Reply
Query
Where is file A?
86

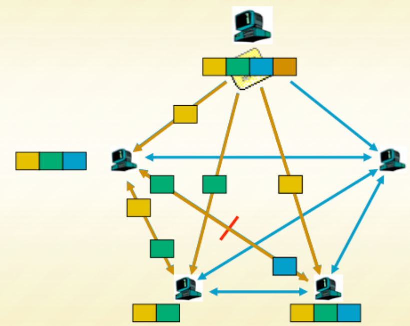

KaZaA: File Insert
insert(X,
123.2.21.23)
...
Publish
I have X!
123.2.21.23
87


KaZaA: File Search
search(A)
-->
123.2.22.50
123.2.22.50
search(A)
-->
123.2.0.18
Query
Replies
Where is file A?
123.2.0.18
88

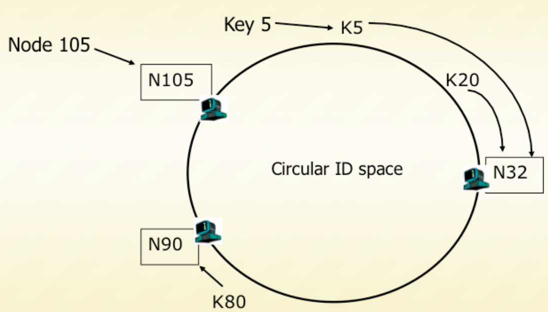
BitTorrent: Publish/Join



BitTorrent: Fetch

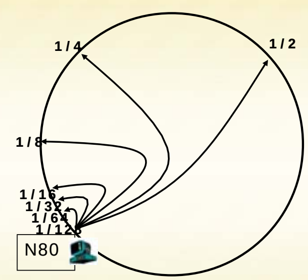


DHT: Consistent Hashing



A key is stored at its successor: node with next higher ID

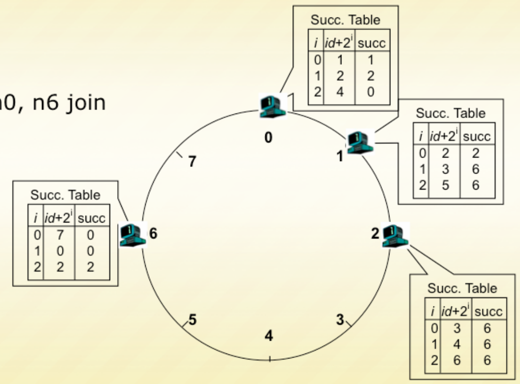
DHT: Chord "Finger Table"



- Entry *i* in the finger table of node *n* is the first node that succeeds or equals $n + 2^i$
- In other words, the ith finger points $1/2^{n-i}$ way around the ring

DHT: Chord Join

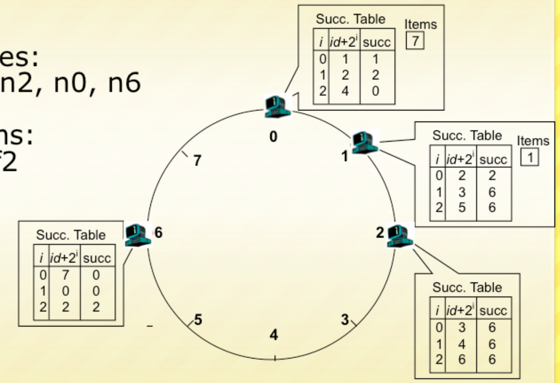
- Nodes n0, n6 join



93

DHT: Chord Join

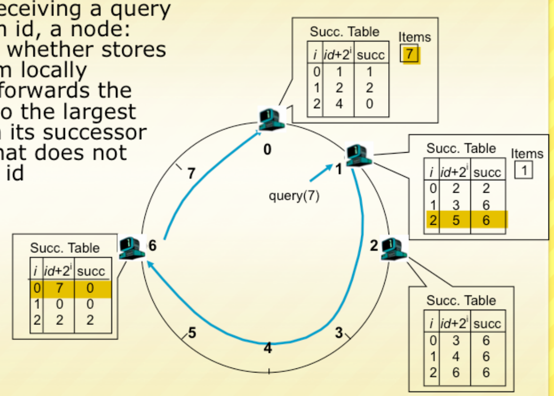
- Nodes:
  n1, n2, n0, n6
- Items:
  f7, f2



94

DHT: Chord Routing

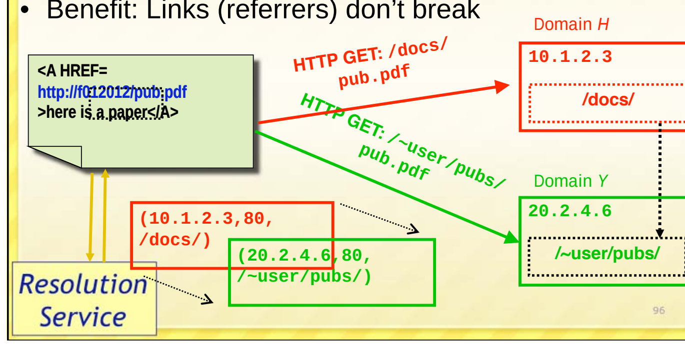
- Upon receiving a query for item id, a node:
- Checks whether stores the item locally
- If not, forwards the query to the largest node in its successor table that does not exceed id



95

Flat Names Example

- SID abstracts all object reachability information
- Objects: any granularity (files, directories)
- Benefit: Links (referrers) don't break



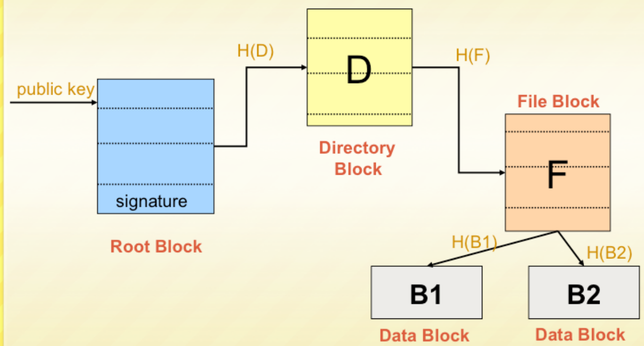
96

P2P-enabled Applications: Self-Certifying Names

- Name = Hash(pubkey, salt)
- Value = <pubkey, salt, data, signature>
  - can verify name related to pubkey and pubkey signed data
- Can receive data from caches or other 3rd parties without worry
  - much more opportunistic data transfer

97

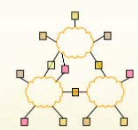
CFS



98

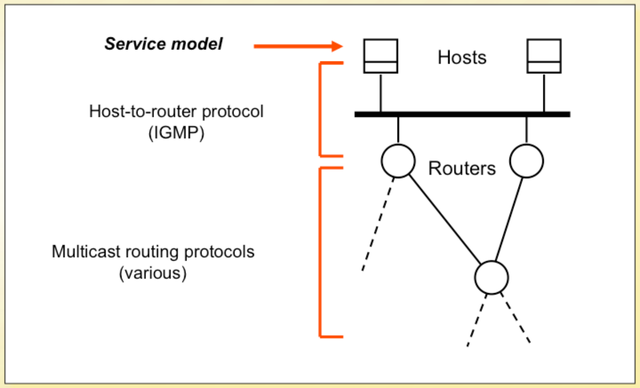
99

15-446 Distributed Systems Spring 2009

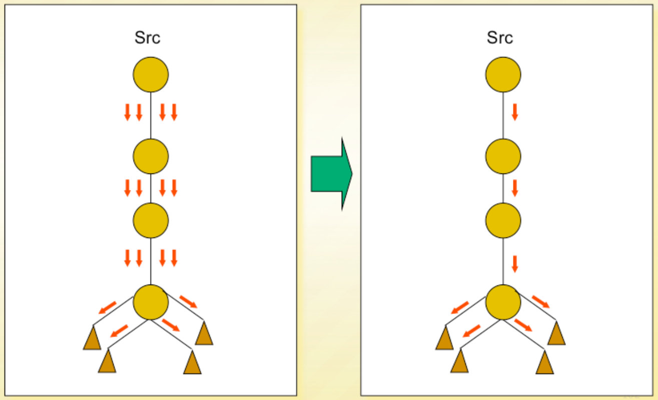


L-20 Multicast

IP Multicast Architecture

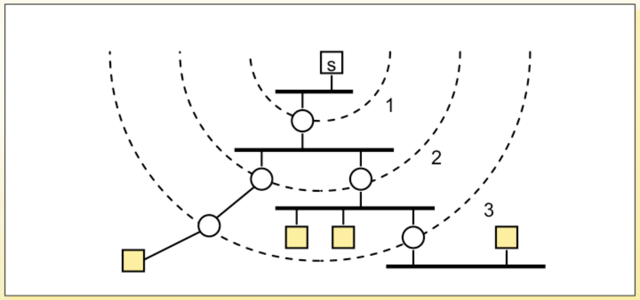


Multicast – Efficient Data Distribution

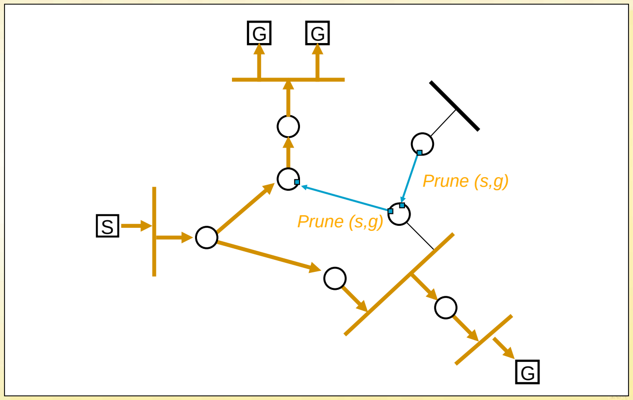


Multicast Scope Control – Small TTLs

- TTL expanding-ring search to reach or find a nearby subset of a group



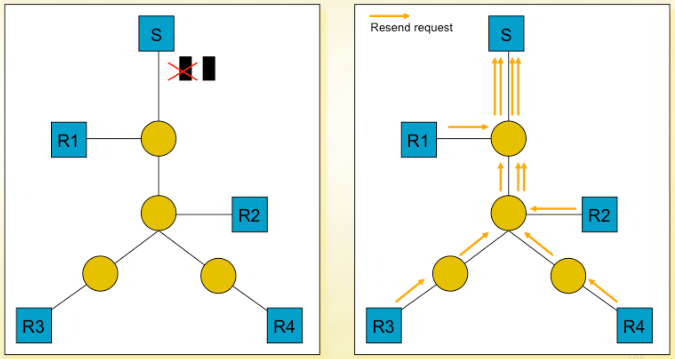
Prune



Implosion

Packet 1 is lost

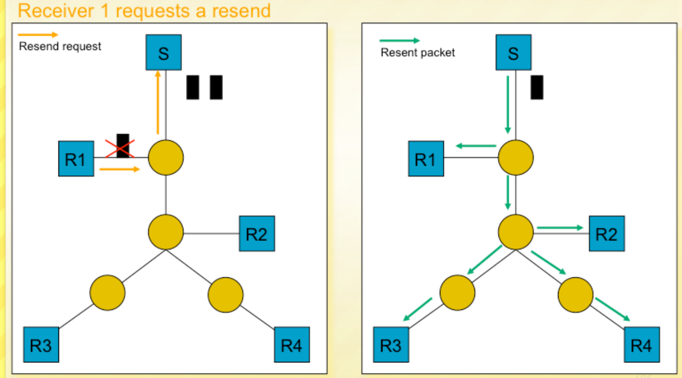
All 4 receivers request a resend



Exposure

Packet 1 does not reach R1;
Receiver 1 requests a resend

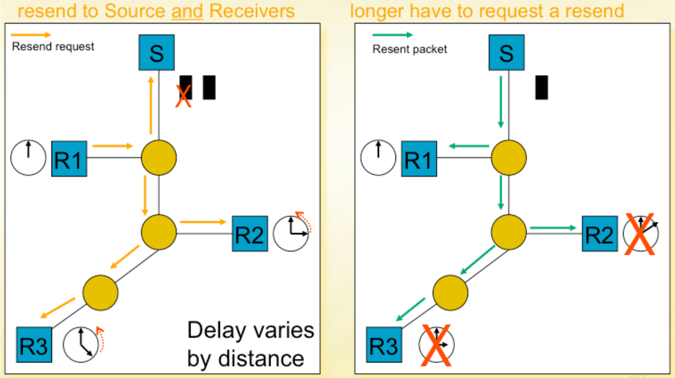
Packet 1 resent to all 4 receivers



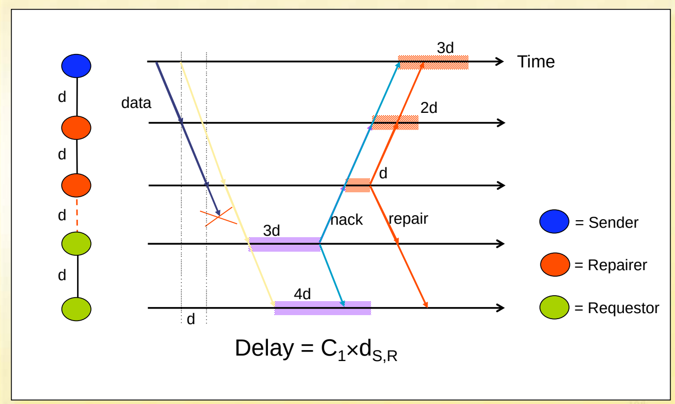
SRM Request Suppression

Packet 1 is lost; R1 requests
resend to Source and Receivers

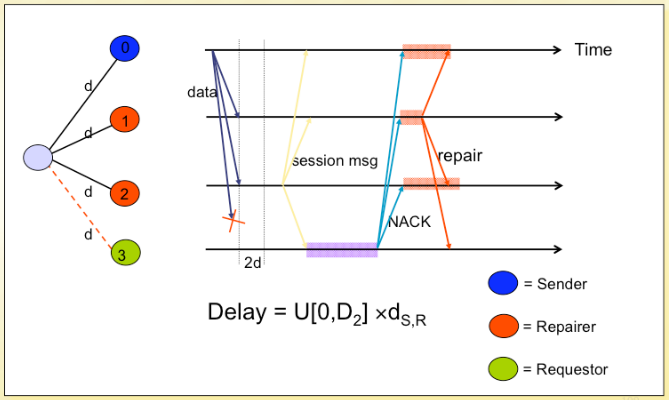
Packet 1 is resent; R2 and R3 no
longer have to request a resend



Deterministic Suppression

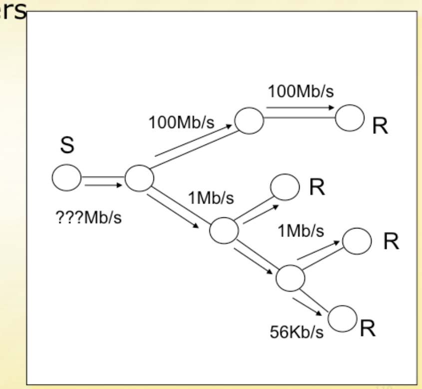


SRM: Stochastic Suppression

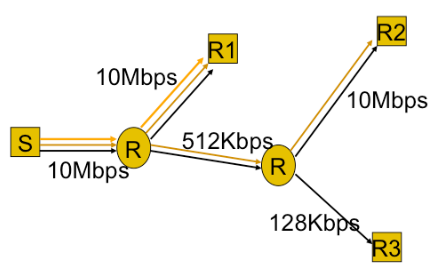


Multicast Congestion Control

- What if receivers have very different bandwidths?
- Send at max?
- Send at min?
- Send at avg?



Layered Media Streams

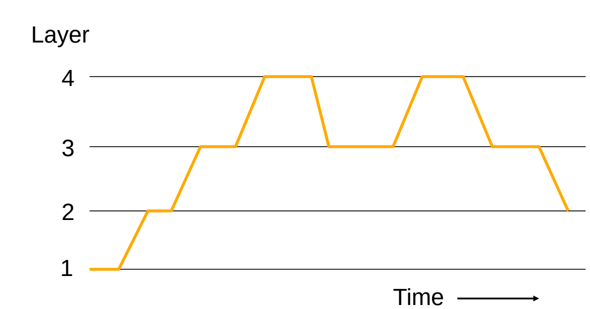


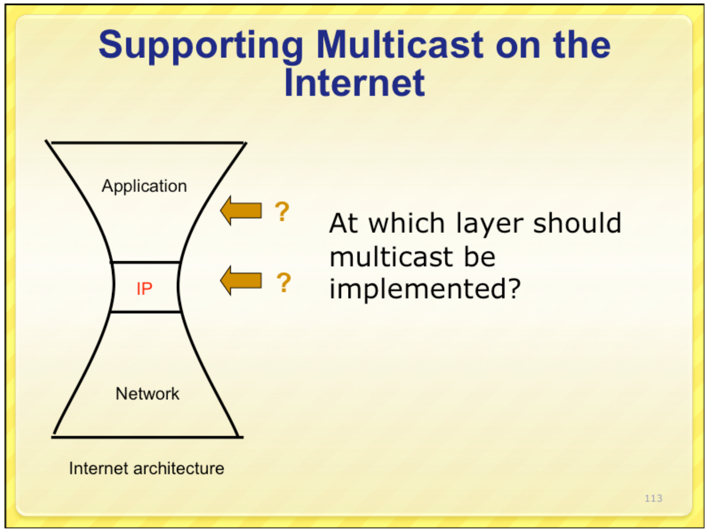
R1 joins layer 1, joins layer 2 joins layer 3

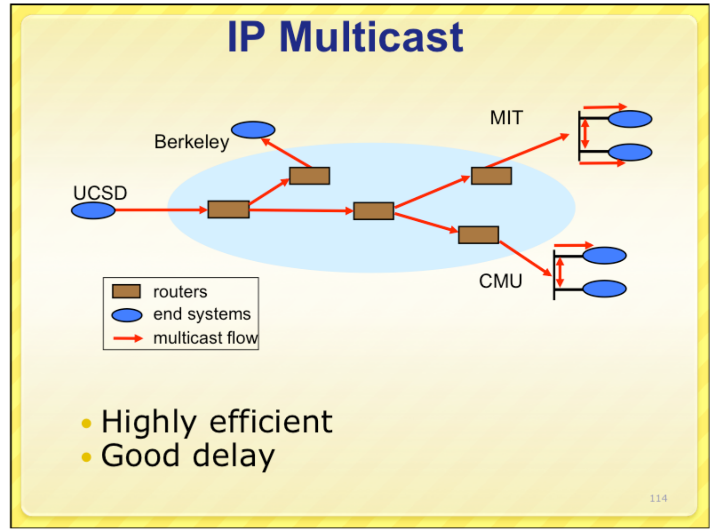
R2 join layer 1, join layer 2 fails at layer 3

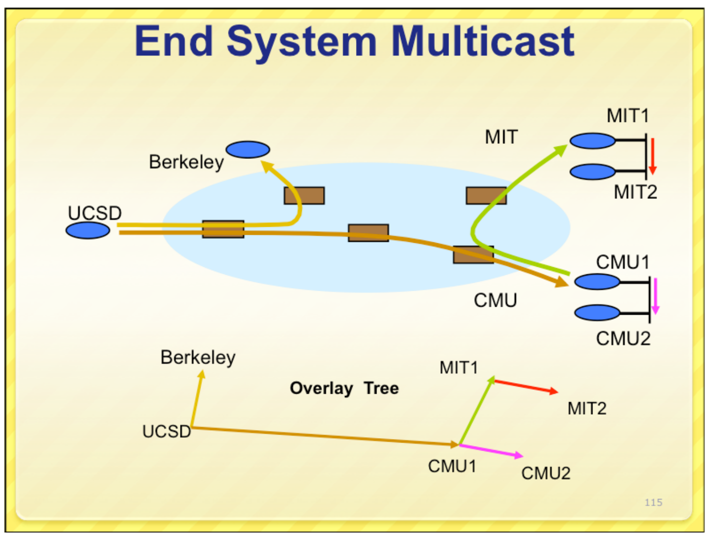
R3 joins layer 1, fails at layer 2

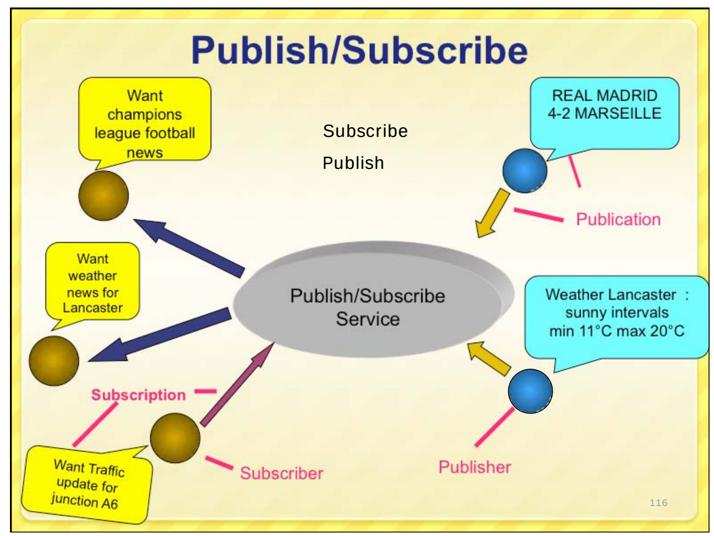
Join Experiments




Supporting Multicast on the Internet
Application
IP
Network
?
?
Internet architecture
At which layer should multicast be implemented?
113


IP Multicast
Berkeley
MIT
UCSD
CMU
routers
end systems
multicast flow
• Highly efficient
• Good delay
114


End System Multicast
MIT1
MIT
Berkeley
MIT2
UCSD
CMU1
CMU
CMU2
Berkeley
Overlay Tree
MIT1
MIT2
UCSD
CMU1
CMU2
115


Publish/Subscribe
Want champions league football news
Subscribe
Publish
REAL MADRID 4-2 MARSEILLE
Publication
Want weather news for Lancaster
Publish/Subscribe Service
Weather Lancaster : sunny intervals min 11°C max 20°C
Subscription
Want Traffic update for junction A6
Subscriber
Publisher
116

Key attributes of P/S communication model

- The publishing entities and subscribing entities are anonymous
- The publishing entities and subscribing entities are highly de-coupled
- Asynchronous communication model
- The number of publishing and subscribing entities can dynamically change without affecting the entire system

117

Subject based vs. Content based

- Subject based:
  - Generally also known as topic based, group based or channel based event filtering.
  - Here each event is published to one of these channels by its publisher
  - A subscriber subscribes to a particular channel and will receive all events published to the subscribed channel.
  - Simple process for matching an event to subscriptions

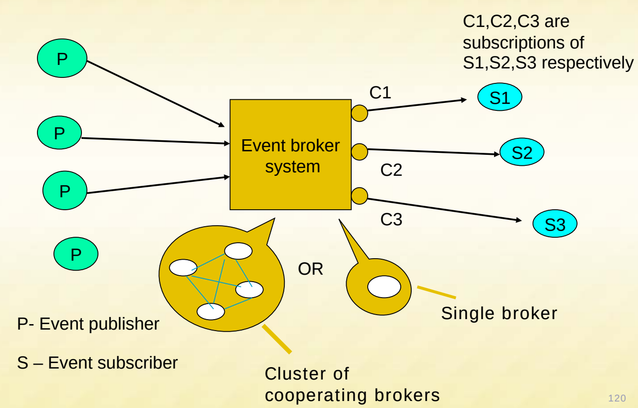
118

Subject based vs. Content based

- Content based:
  - More flexibility and power to subscribers, by allowing to express as an arbitrary query over the contents of the event.
  - E.g. Notify me of all stock quotes of IBM from New York stock exchange if the price is greater than 150
  - Added complexity in matching an event to subscriptions

119

Event routing



120

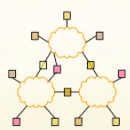
Basic elements of P/S model

- Event data model
  - Structure
  - Types
- Subscription model
  - Filter language
  - Scope (subject, content, context)
- General challenge
  - Expressiveness vs. Scalability

121

122

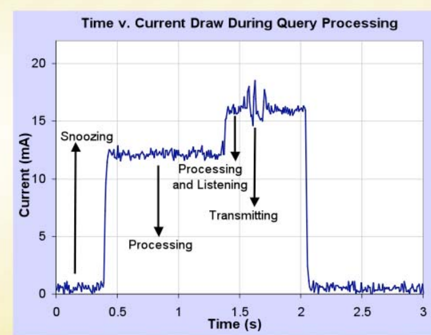
15-446 Distributed Systems Spring 2009



L-22 Sensor Networks

Metric: Communication

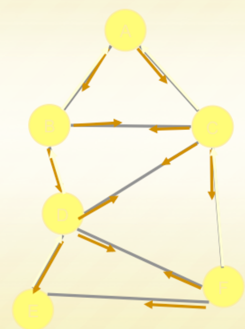
- Lifetime from one pair of AA batteries
  - 2-3 days at full power
  - 6 months at 2% duty cycle
- Communication dominates cost
  - < few mS to compute
  - 30mS to send message



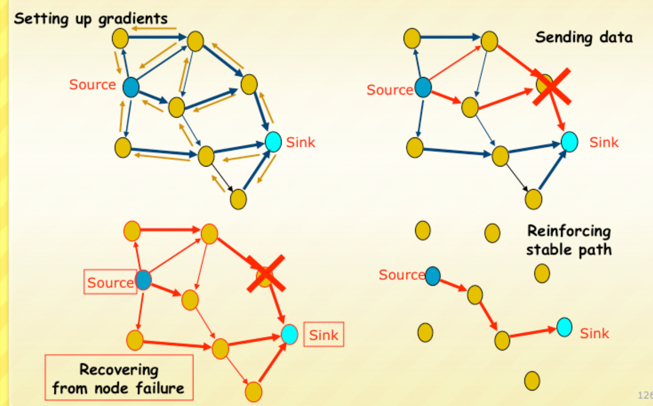
124

Communication In Sensor Nets

- Radio communication has high link-level losses
  - typically about 20% @ 5m
- Ad-hoc neighbor discovery
- Tree-based routing



Illustrating Directed Diffusion



Summary

- Data Centric
  - Sensors net is queried for specific data
  - Source of data is irrelevant
  - No sensor-specific query
- Application Specific
  - In-sensor processing to reduce data transmitted
  - In-sensor caching
- Localized Algorithms
  - Maintain minimum local connectivity – save energy
  - Achieve global objective through local coordination
- Its gains due to aggregation and duplicate suppression may make it more viable than ad-hoc routing in sensor networks

Basic Aggregation

- In each epoch:
  - Each node samples local sensors once
  - Generates **partial state record (PSR)**
    - local readings
    - readings from children
  - Outputs PSR during its comm. slot.
- At end of epoch, PSR for whole network output at root
- (In paper: pipelining, grouping)

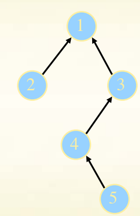
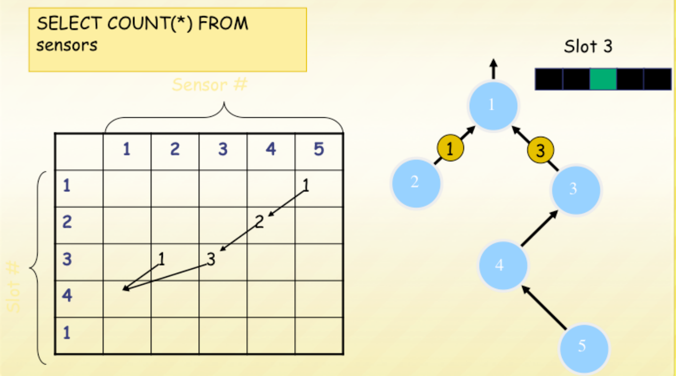


Illustration: Aggregation



Taxonomy of Aggregates

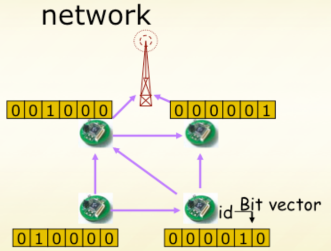
- TAG insight: classify aggregates according to various functional properties
  - Yields a general set of optimizations that can automatically be applied

Property	Examples	Affects
Partial State	MEDIAN : unbounded, MAX : 1 record	Effectiveness of TAG
Duplicate Sensitivity	MIN : dup. insensitive, AVG : dup. sensitive	Routing Redundancy
Exemplary vs. Summary	MAX : exemplary COUNT: summary	Applicability of Sampling, Effect of Loss
Monotonic	COUNT : monotonic AVG : non-monotonic	Hypothesis Testing, Snooping

130

A Naïve ODI Algorithm

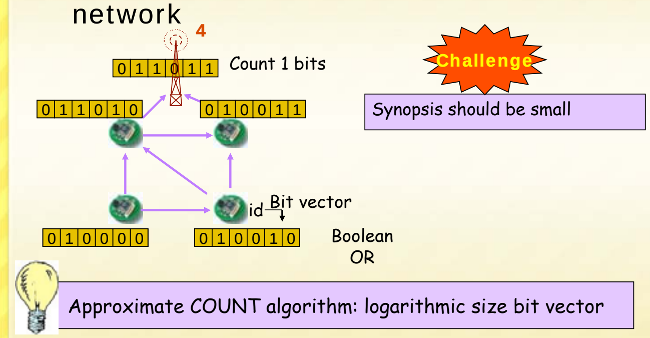
- Goal: count the live sensors in the network



131

Synopsis Diffusion (SenSys'04)

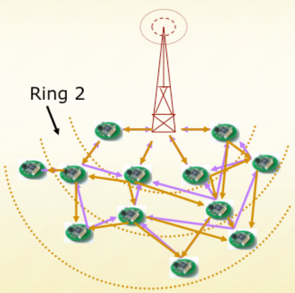
- Goal: count the live sensors in the network



132

Synopsis Diffusion over Rings

- A node is in ring *i* if it is *i* hops away from the base-station
- <u>Broadcasts</u> by nodes in ring *i* are received by neighbors in ring *i*-1
- Each node transmits once = optimal energy cost (same as Tree)



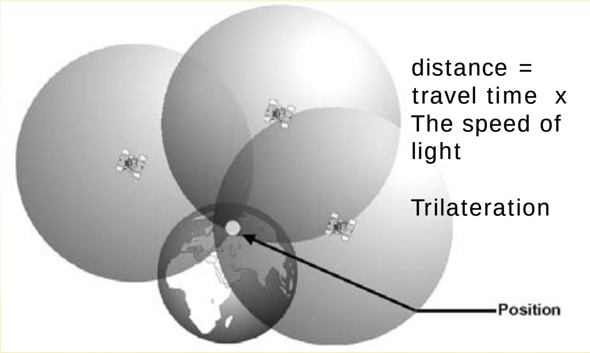
133

134

15-446 Distributed Systems Spring 2009

Localization

Positioning with synchronized clock



Accounting for the clock offset

- Satellites' clocks are well synchronized
- Receiver clock is not synchronized.
- Need to estimate 4 unknowns
  - (x, y, z, Δt)
  - Δt is the clock offset of the receiver
  - R: real distance, PSR : estimated distance
  - R = PSR - Δt ·c

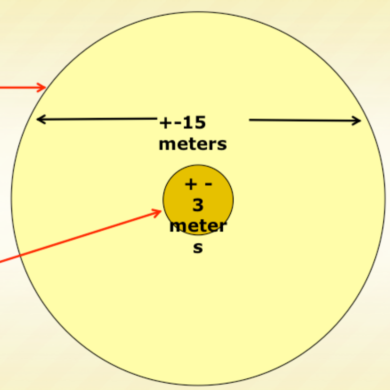
$$R = \sqrt{(X_{Sat} - X_{User})^2 + (Y_{Sat} - Y_{User})^2 + (Z_{Sat} - Z_{User})^2}$$

Wide Area Augmentation System

- Error correction system that uses reference ground stations
- 25 reference stations in US
- Monitor GPS and send correction values to two geo-stationary satellites
- The two geo-stationary satellites broadcast back to Earth on GPS L1 frequency (1575.42MHz)
- Only available in North America, WASS enabled GPS receiver needed

How good is WAAS?

With Selective Availability set to zero, and under ideal conditions, a GPS receiver without WAAS can achieve fifteen meter accuracy most of the time.*



Under ideal conditions a WAAS equipped GPS receiver can achieve three meter accuracy 95% of the time.*

* Precision depends on good satellite geometry, open sky view, and no user induced errors.

WiFi Positioning System

- Exploit wide-scale WiFi deployment
  - WiFi density increasing to point of overlap
- WiFi base stations broadcast unique IDs
- Position WiFi devices using a map of AP's MAC to location

Place Lab 140

Indoor localization system

- Usually more fine grained localization needed
  - Often 3D (2.5D) : x,y and floor
  - Often want to locate users in an office
- RADAR
  - Trilateration based on signal strength from APs
  - Hard to predict distance based on signal strength because signal is blocked by walls and structures
  - Use site-surveying
- Lots of research has been done
  - MIT Cricket (RF + ultrasound)
  - AeroScout (WiFi), Ekahau (WiFi)

IP-Geography Mapping

- Goal: Infer the geographic location of an Internet host given its IP address.
- Why is this interesting?
  - enables location-aware applications
  - example applications:
    - Territorial Rights Management
    - Targeted Advertising
    - Network Diagnostics
- Why is this hard?
  - IP address does not inherently indicate location
  - proxies hide client identity, limit visibility into ISPs
- Desirable features of a solution
  - easily deployable, accuracy, confidence indicator

142

IP2Geo

- Infer geo-location of IP based on various "properties"
  - DNS names of routers often indicate location
  - Network delay correlates with geographic distance
  - Subnets are clustered
- Three techniques
  - GeoTrack
  - GeoPing
  - GeoClusters

IP2Geo Conclusions

- IP2Geo encompasses a diverse set of techniques
  - GeoTrack: DNS names
  - GeoPing: network delay
  - GeoCluster: geographic clusters
- Median error 20-400 km
  - GeoCluster also provides confidence indicator
- Each technique best suited for a different purpose
  - GeoTrack: locating routers, tracing geographic path
  - GeoPing: location determination for proximity-based routing (e.g., CoopNet)
  - GeoCluster: best suited for location-based services
- Publications at SIGCOMM 2001 & USENIX 2002

144

GeoTrack

- Location info often embedded in router DNS names
  - ngcore1-serial8-0-0-0.Seattle.cw.net, 184.atm6-0.xr2.ewr1.alter.net
- GeoTrack operation
  - do a traceroute to the target IP address
  - determine location of last recognizable router along the path
- Key ideas in GeoTrack
  - partitioned city code database to minimize chance of false match
  - ISP-specific parsing rules
  - delay-based correction
- Limitations
  - routers may not respond to traceroute
  - DNS name may not contain location information or lookup may fail
  - target host may be behind a proxy or a firewall

GeoPing

- Nearest Neighbor in Delay Space(NNDS)
  - delay vector: delay measurements from a host to a fixed set of landmarks
  - delay map: database of delay vectors and locations for a set of known hosts
    (50,45,20,35) ↔ Indianapolis, IN
    (10,20,40,60) ↔ Seattle, WA
    …
  - target location corresponds to best match in delay map
  - optimal dimensionality of delay vector is 7-9

146

GeoCluster

- Basic Idea: identify geographic clusters
  - partial IP-location database
    - construct a database of the form (IPaddr, likely location)
    - partial in coverage and potentially inaccurate
    - sources: HotMail registration/login logs, TVGuide query logs
  - cluster identification
    - use prefix info. from BGP tables to identify topological clusters
    - assign each cluster a location based on IP-location database
    - do sub-clustering when no consensus on a cluster's location
  - location of target IP address is that of best matching cluster

147

148

15-446 Distributed Systems Spring 2009

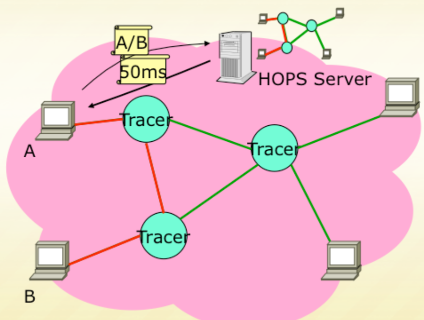
L-24 Adaptive Applications

Revisit: Why is Automated Adaptation Hard?

- Must infer Internet performance
  - Scalability
  - Accuracy
  - Tradeoff with timeliness
- Support for a variety of applications
  - Different performance metrics
  - API requirements
- Layered implementations hide information

State of the Art: IDMaps [Francis et al '99]

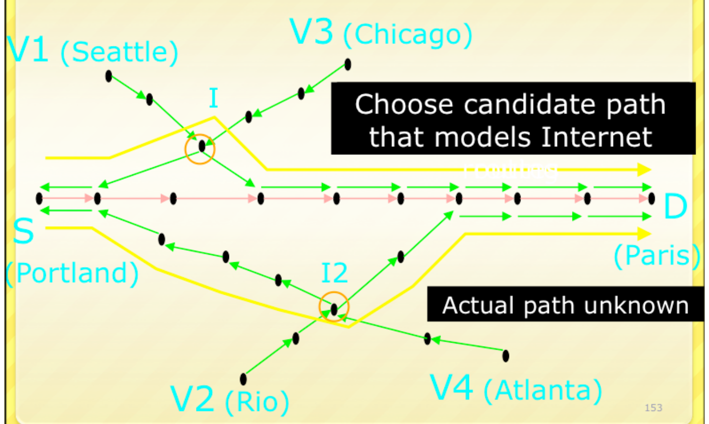
- A network distance prediction service



iPlane: Build a Structural Atlas of the Internet

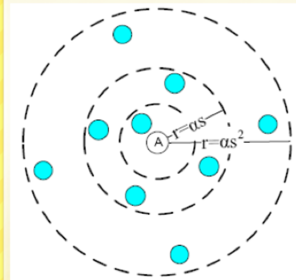
- Use PlanetLab + public traceroute servers
  - Over 700 geographically distributed vantage points
- Build an atlas of Internet routes
  - Perform traceroutes to a random sample of BGP prefixes
  - Cluster interfaces into PoPs
  - Repeat daily from vantage points

Model for Path Prediction



153

Multi-resolution Ring structure



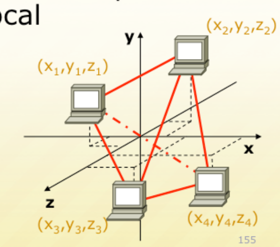
For the i^{th} ring:
- Inner Radius $r_i = \alpha s^{i-1}$
- Outer Radius $R_i = \alpha s^i$

α is a constant
s is multiplicative increase factor
$r_0 = 0$, $R_0 = \alpha$
Each node keeps track of finite rings

154

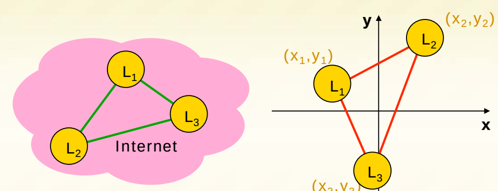
New Fundamental Concept: "Internet Position"

- Using GNP, every host can have an "Internet position"
  - O(N) positions, as opposed to $O(N^2)$ distances
- Accurate network distance estimates can be rapidly computed from "Internet positions"
- "Internet position" is a local property that can be determined **before** applications need it
- Can be an interface for independent systems to interact



155

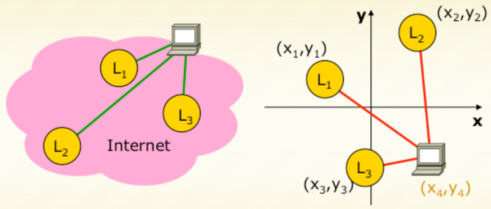
Landmark Operations (Basic Design)



- Measure inter-Landmark distances
  - Use minimum of several round-trip time (RTT) samples
- Compute coordinates by minimizing the discrepancy between measured distances and geometric distances
  - Cast as a generic multi-dimensional minimization problem, solved by a central node

156

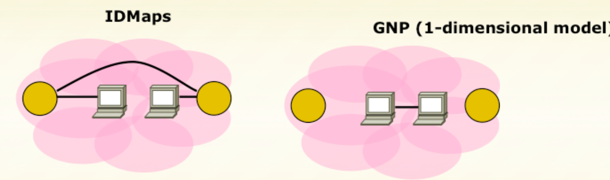
Ordinary Host Operations (Basic Design)



- Each host measures its distances to all the Landmarks
- Compute coordinates by minimizing the discrepancy between measured distances and geometric distances
  - Cast as a generic multi-dimensional minimization problem, solved by each host

157

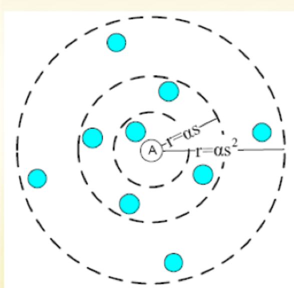
Why the Difference?



- IDMaps overpredicts

158

Multi-resolution Ring structure

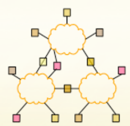


For the i^{th} ring:
  Inner Radius $r_i = \alpha s^{i-1}$
  Outer Radius $R_i = \alpha s^i$
α is a constant
s is multiplicative increase factor
$r_0 = 0$, $R_0 = \alpha$
Each node keeps track of finite rings

159

160

15-446 Distributed Systems Spring 2009



L-25 Cluster Computing

GFS: Architecture

- One master server (state replicated on backups)
- Many chunk servers (100s – 1000s)
  - Spread across racks; intra-rack b/w greater than inter-rack
  - Chunk: 64 MB portion of file, identified by 64-bit, globally unique ID
- Many clients accessing same and different files stored on same cluster

162

GFS: Architecture (2)

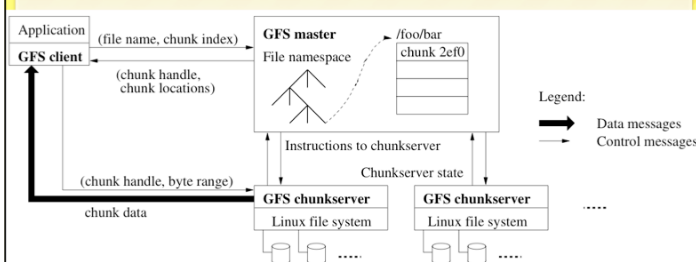


Figure 1: GFS Architecture

163

Master Server

- Holds all metadata:
  - Namespace (directory hierarchy)
  - Access control information (per-file)
  - Mapping from files to chunks
  - Current locations of chunks (chunkservers)
- Delegates consistency management
- Garbage collects orphaned chunks
- Migrates chunks between chunkservers

Holds all metadata in RAM; very fast operations on file system metadata

164

Client Read

- Client sends master:
  - read(file name, chunk index)
- Master's reply:
  - chunk ID, chunk version number, locations of replicas
- Client sends "closest" chunkserver w/ replica:
  - read(chunk ID, byte range)
  - "Closest" determined by IP address on simple rack-based network topology
- Chunkserver replies with data

165

Client Write (2)

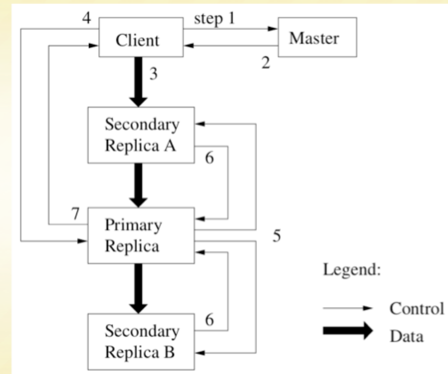


Figure 2: Write Control and Data Flow

166

Client Record Append

- Google uses large files as queues between multiple producers and consumers
- Same control flow as for writes, except…
- Client pushes data to replicas of last chunk of file
- Client sends request to primary
- Common case: request fits in current last chunk:
  - Primary appends data to own replica
  - Primary tells secondaries to do same at same byte offset in theirs
  - Primary replies with success to client

167

GFS: Consistency Model (2)

- Changes to data are ordered as chosen by a primary
  - All replicas will be consistent
  - But multiple writes from the same client may be interleaved or overwritten by concurrent operations from other clients
- Record append completes at least once, at offset of GFS's choosing
  - Applications must cope with possible duplicates

168

What If the Master Reboots?

- Replays log from disk
  - Recovers namespace (directory) information
  - Recovers file-to-chunk-ID mapping
- Asks chunkservers which chunks they hold
  - Recovers chunk-ID-to-chunkserver mapping
- If chunk server has older chunk, it's stale
  - Chunk server down at lease renewal
- If chunk server has newer chunk, adopt its version number
  - Master may have failed while granting lease

169

What if Chunkserver Fails?

- Master notices missing heartbeats
- Master decrements count of replicas for all chunks on dead chunkserver
- Master re-replicates chunks missing replicas in background
  - Highest priority for chunks missing greatest number of replicas

170

GFS: Summary

- Success: used actively by Google to support search service and other applications
  - Availability and recoverability on cheap hardware
  - High throughput by decoupling control and data
  - Supports massive data sets and concurrent appends
- Semantics not transparent to apps
  - Must verify file contents to avoid inconsistent regions, repeated appends (at-least-once semantics)
- Performance not good for all apps
  - Assumes read-once, write-once workload (no client caching!)

171

MapReduce Programming Model

- Input & Output: sets of <key, value> pairs
- Programmer writes 2 functions:

```
map (in_key, in_value) → list(out_key,
  intermediate_value)
```

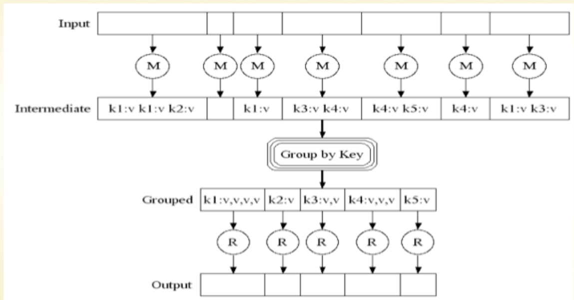
  - Processes <k,v> pairs
  - Produces intermediate pairs

```
reduce (out_key, list(interm_val)) →
  list(out_value)
```

  - Combines intermediate values for a key
  - Produces a merged set of outputs (may be also <k,v> pairs)

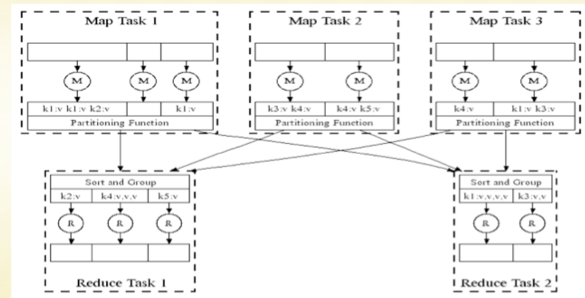
172

MapReduce: Example



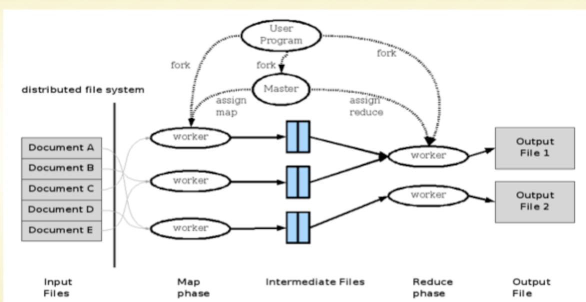
173

MapReduce in Parallel: Example



174

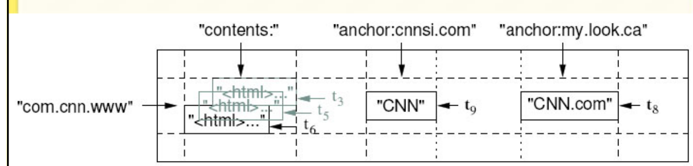
MapReduce: Execution overview



175

BigTable: Basic Data Model

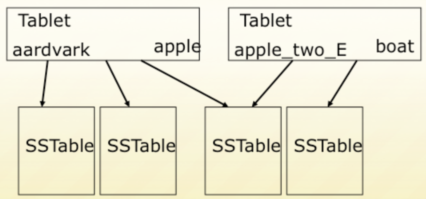
- A BigTable is a sparse, distributed persistent multi-dimensional sorted map *(row, column, timestamp) → cell contents*



176

Table

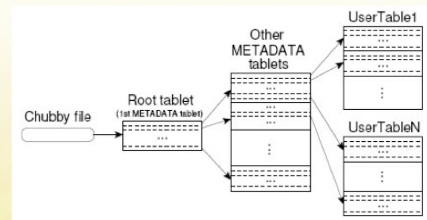
- Multiple tablets make up the table
- SSTables can be shared
- Tablets do not overlap, SSTables can overlap



177

Tablet Location

- Since tablets move around from server to server, given a row, how do clients find the right machine?
  - Need to find tablet whose row range covers the target row



178

Chubby

- {lock/file/name} service
- Coarse-grained locks, can store small amount of data in a lock
- 5 replicas, need a majority vote to be active
- Also an OSDI '06 Paper

179

Master's Tasks

- Use Chubby to monitor health of tablet servers, restart failed servers
  - Tablet server registers itself by getting a lock in a specific directory chubby
    - Chubby gives "lease" on lock, must be renewed periodically
    - Server loses lock if it gets disconnected
  - Master monitors this directory to find which servers exist/are alive
    - If server not contactable/has lost lock, master grabs lock and reassigns tablets
    - GFS replicates data. Prefer to start tablet server on same machine that the data is already at

180

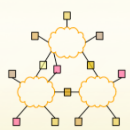
Master's Tasks (Cont)

- When (new) master starts
  - grabs master lock on chubby
    - Ensures only one master at a time
  - Finds live servers (scan chubby directory)
  - Communicates with servers to find assigned tablets
  - Scans metadata table to find all tablets
    - Keeps track of unassigned tablets, assigns them
    - Metadata root from chubby, other metadata tablets assigned before scanning.

181

182

15-446 Distributed Systems Spring 2009



L-26 Cluster Computer
(borrowed from Randy Katz, UCB)

Energy Proportional Computing

"The Case for Energy-Proportional Computing," Luiz André Barroso, Urs Hölzle, *IEEE Computer* December 2007

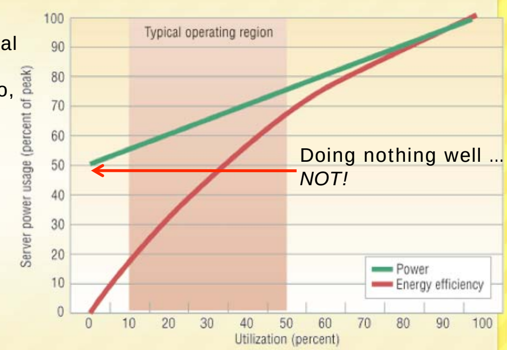


Figure 2. Server power usage and energy efficiency at varying utilization levels, from idle to peak performance. Even an energy-efficient server still consumes about half its full power when doing virtually no work.

184

Energy Proportional Computing

"The Case for Energy-Proportional Computing," Luiz André Barroso, Urs Hölzle, *IEEE Computer* December 2007

It is surprisingly hard to achieve high levels of utilization of typical servers (and your home PC or laptop is even worse)

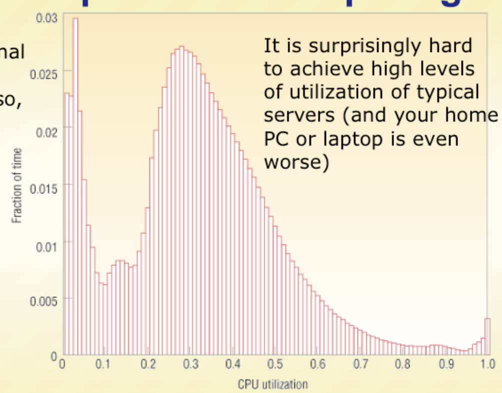
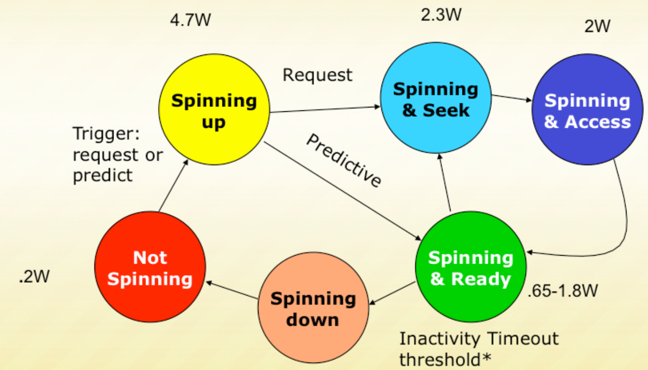


Figure 1. Average CPU utilization of more than 5,000 servers during a six-month period. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum

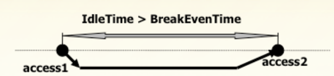
185

Spin-down Disk Model

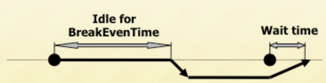


Disk Spindown

- Disk Power Management – Oracle (off-line)



- Disk Power Management – Practical scheme (on-line)



187

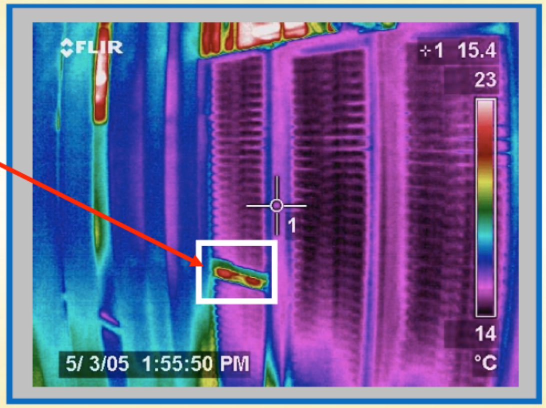
Source: from the presentation slides of the authors

Spin-Down Policies

- Fixed Thresholds
  - T_{out} = spin-down cost s.t. $2 * E_{transition} = P_{spin} * T_{out}$
- Adaptive Thresholds: T_{out} = f (recent accesses)
  - Exploit burstiness in T_{idle}
- Minimizing Bumps (user annoyance/latency)
  - Predictive spin-ups
- Changing access patterns (making burstiness)
  - Caching
  - Prefetching

Thermal Image of Typical Cluster

Rack
Switch



M. K. Patterson, A. Pratt, P. Kumar,
"From UPS to Silicon: an end-to-end evaluation of datacenter efficiency", Intel Corporation

189

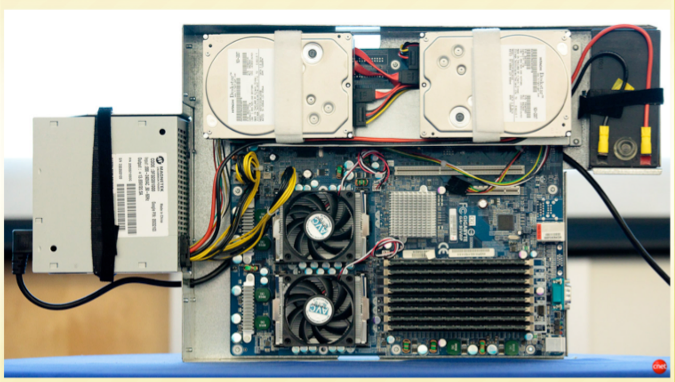
Datacenter Power Efficiencies

- Power conversions in server
  - Power supply (<80% efficiency)
  - Voltage regulation modules (80% common)
  - Better available (95%) and inexpensive
- Simple rules to minimize power distribution losses in priority order
  1. Avoid conversions (indirect UPS or no UPS)
  2. Increase efficiency of conversions
  3. High voltage as close to load as possible
  4. Size board voltage regulators to load and use high quality
  5. Direct Current small potential win (but regulatory issues)
- Two interesting approaches:
  - 480VAC to rack and 48VDC (or 12VDC) within rack
  - 480VAC to PDU and 277VAC (1 leg of 480VAC 3-phase distribution) to each server

190

James Hamilton, Amazon

Google 1U + UPS



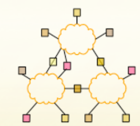
191

Why built-in batteries?

- Building the power supply into the server is cheaper and means costs are matched directly to the number of servers
- Large UPSs can reach 92 to 95 percent efficiency vs. 99.9 percent efficiency for server mounted batteries

192

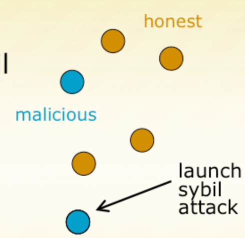
15-446 Distributed Systems Spring 2009



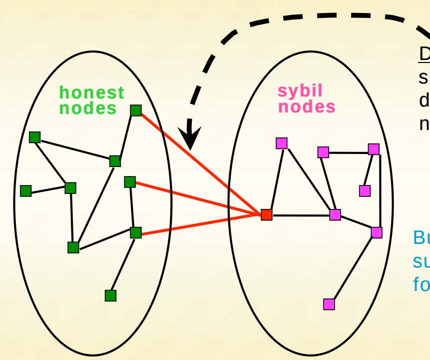
L-27 Social Networks and Other Stuff

Background: Sybil Attack

- Sybil attack: Single user pretends many fake/sybil identities
  - Creating multiple accounts from different IP addresses
- Sybil identities can become a large fraction of all identities
  - Out-vote honest users in collaborative tasks



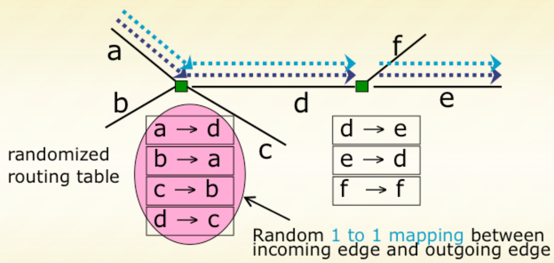
SybilGuard Basic Insight



<u>Dis-proportionally</u> small cut disconnecting a large number of identities

But cannot search for such cut brute-force…

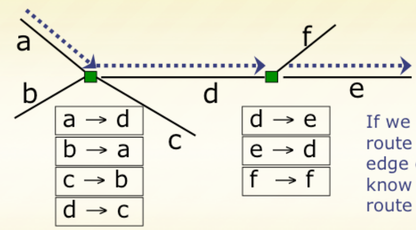
Random Route: Convergence



- Using routing table gives Convergence Property
  - Routes merge if crossing the same edge

197

Random Route: Back-traceable

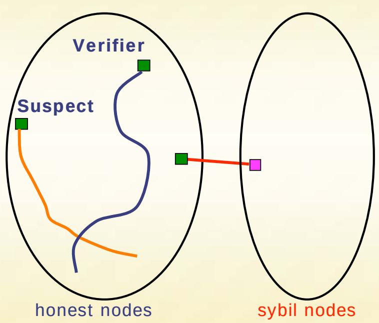


If we know the route traverses edge e, then we know the whole route

- Using 1-1 mapping gives Back-traceable Property
  - Routes may be back-traced

198

Random Route Intersection: Honest Nodes



- Verifier accepts a suspect if the two routes intersect
  - Route length w:

    $\sim \sqrt{n} \log n$
  - W.h.p., verifier's route stays within honest region
  - W.h.p., routes from two honest nodes intersect

199