

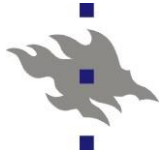


HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Peer-to-Peer and Grid Computing

Chapter 7: Other Issues





Chapter Outline

- n Further issues in P2P systems
- n Security (in DHTs)
 - n Overview of problems
 - n Sybil attack
- n Privacy and anonymity
 - n Can these be protected?
- n Napster legal case
 - n Why original Napster failed and what can we learn?
- n Online music stores
 - n Alternative to file sharing?



Security in DHTs

- n DHT architectures assumes a trusted system
 - n True in corporate environments, but not on the Internet
- n One solution: Central certificate-granting authority
 - n Used by Pastry and its related projects
 - n Constrains membership in DHT

- n One attack: Return incorrect data
 - n Easy to avoid through cryptographic techniques
 - n Detect and ignore non-authentic data

- n Focus: **Attacks that prevent participants from finding the data**
 - n Threatens the liveliness of the system



DHT Components

DHTs have following components:

1. Key identifier space
 2. Node identifier space
 3. Rules for associating keys to nodes
 4. Per-node routing tables that refer to other nodes
 5. Rules for updating routing tables as nodes join and leave
- Any of the above may be the target of the attack



Adversary Model

- n Adversaries are participants in DHT that do not follow protocol correctly

Assumptions:

- n Malicious node can generate arbitrary packets

 - n Includes forged source IP address

- n Can receive only packets addressed to itself

 - n Not able to overhear communications between other nodes

- n Malicious nodes can conspire together, but still limited as above



Types of Attacks

1. Routing attacks
 2. Attack against data storage
 3. Miscellaneous attacks
- n First goal: **Detect attack**
 - n Violation of invariants or contracts
 - n What to do when an attack is detected?
 - n Is other node malicious?
 - n Did other node simply not detect attack?
 - n Achieving **verifiability** is vital



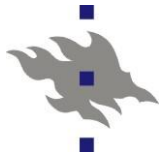
Routing Attacks

- n Routing is responsible for maintaining routing tables and sending messages to correct nodes
- n Routing **must** function correctly
 - n Define invariants and check them
- n Attacker can forward messages incorrectly
 - n But: Each hop should get “closer” to destination
 - n Querying node should check this
 - n Allow querying node to observe lookup process
 - For example, processing messages recursively hides this
- n Attacker can claim wrong node is responsible node
 - n Querying node is “far away”, cannot verify this
 - n Assign keys to nodes in a verifiable way
 - n Often: Assign node IDs in a verifiable way (e.g., IP address)
 - For example, CAN lets node pick its own ID...



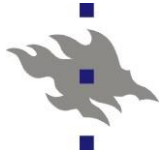
More Routing Attacks

- n Attacker sends incorrect routing updates
 - n Blatantly wrong updates can be detected
 - n If DHT allows several choices for next hop
 - Attacker can pick a “bad” node
 - Not necessarily a problem with correctness, only performance
 - Can be a problem for some applications (anonymity)
 - n Server selection can be abused



More Routing Attacks 2

- n Attacker can partition network
 - n If new node contacts attacker first, attacker can partition network (can even hijack nodes from real network)
 - n Parallel network is consistent and “looks OK”
 - Attacker can track nodes
 - n Bootstrap from a trusted source: Hard to get in dynamic networks, public keys might help
 - n Cross check routing tables with random queries
 - Assumes we were part of network earlier, still not totally safe



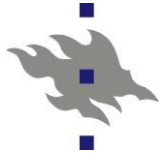
Storage and Retrieval Attacks

- n Attacker can deny existence of data
 - n Or return wrong data
- n Must implement replication at storage layer
 - n Who creates replicas?
 - n Clients must be able to verify that all copies were created
- n **Avoid single points of responsibility**
 - n Replication with multiple hash functions is one good way
- n Big problem if system does not verify IDs
 - n Any node can become responsible for any data
 - n For example, Chord allows virtual nodes



Miscellaneous Attacks

- n Attacker can behave inconsistently
 - n Some nodes see it as good, others as bad
 - n Maintain good face to nearby nodes
 - n How would a distant node convince neighbors of bad node?
 - Public keys and signatures could solve this
- n Denial of service
 - n Attacker floods a node with messages
 - n Node appears failed to the rest of the network
 - n Replication helps, but attacker may succeed if replication not sufficient
 - n Replicas should be in physically different locations
 - DHT assigns keys to nodes randomly, should be OK
 - Large attacks require lot of resources



More Miscellaneous Attacks

- n Attacker can join and leave the network rapidly
 - n Causes lot of stabilization traffic in network
 - n Loss of performance, maybe loss of correctness
 - n Works well if stabilization requires lot of data transfer
 - For example, copying of large objects from node to node
 - n DHT must handle this case anyway
- n Attacker can send unsolicited messages
 - n *Q* asks *E* and gets referred to *A*
 - n *E* knows *Q* expects an answer from *A*
 - n *E* forges message from *A* to *Q*
 - n Public keys and signatures (heavy solution)
 - n Random nonce in a message works also



Design Principles

Summary of design principles for secure DHT:

1. Define verifiable system invariants (and verify them!)
2. Allow querying node to observe lookup process
3. Assign keys to nodes in a verifiable way
4. Server selection in routing may be abused
5. Cross-check routing tables with random queries
6. Avoid single points of responsibility



Sybil Attack

n Sybil?

- n From book/movie telling the story of Sybil Isabel Dorsett who suffered from multiple personality disorder

n How to protect against malicious peers?

n For example, data replication

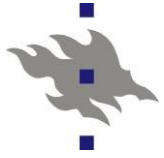
- n A single copy might be on a malicious peer
- n But several copies on different peers are safe, right?

n How can we know that the “different” peers are really different and distinct physical entities?

n Answer: We need a centralized, trusted entity (e.g., CA)

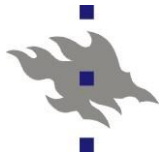
n Without central authority, the problem is *unsolvable*

- n Can be proven mathematically to be unsolvable



What Is The Problem?

- n **Entity:** Real-world entity, e.g., one user
- n **Identity:** Representation of an entity in system
- n Redundancy requires resources to be spread across several entities
 - n Peer-to-peer systems work only with identities
- n How to ensure one entity does not create multiple identities and attack the system that way?
- n This is called the **Sybil Attack**
- n Only solution is a (logically) centralized authority for managing entity-identity mappings



Examples of Solutions

- n Actually centralized authorities:
 - n Certification Authorities, e.g., VeriSign
- n Logically centralized authorities:
 - n Hashing IP address to get DHT identifier (e.g., CFS)
 - n Add host identifiers to DNS names (SFS)
 - n Cryptographic keys in hardware (EMBASSY)
 - n These appear distributed, but they all rely on some centralized authority (e.g., ICANN gives out IP addresses and DNS names)
- n Identities vouching for other identities
 - n For example, PGP web of trust for humans
 - n **NOT a solution!**
 - n Attacker can attack the system early and compromise generation of identities and break chain of vouchers



Results

- n Entity should accept identities only if they have been validated by central authority, itself, or others
 - n In a fully distributed system, only entity itself and others
- n Following can be shown under reasonably realistic assumptions for direct validation:
 1. Even when severely resource constrained, a faulty entity can counterfeit a constant number of multiple identities
 2. Each correct entity must simultaneously validate all the identities it is presented; otherwise, a faulty entity can counterfeit an unbounded number of entities
 - Similar results hold for indirect validation by others
- n **What resources can be used in identification?**
 - n Communication, CPU, storage



Resources as Proof

n Communication

- n Broadcast request for others to identify themselves and accept only responses which come within a certain time interval
- n Model had assumed broadcast communications

n CPU

- n Require other peer to perform some computationally intensive, but easily verifiable, task
- n This requires simultaneous identification (point 2 from above)

n Storage

- n Have others store some uncompressible data and periodically ask them to give back a small piece
- n Would eventually catch a Sybil attack
- n Problem: No storage space left for doing any real work...



Implications of Sybil Attack

- n Need centralized authority for managing identities
- n Logically centralized systems should be aware of their potential (future) vulnerabilities
 - n For example, privacy extensions for IPv6 might break CFS
- n Sybil attack can be avoided under the assumptions:
 - n All entities operate under identical resource constraints
 - n All presented identities are validated simultaneously by all entities, coordinated over the whole system
 - n For indirect validation, the number of vouchers must exceed the number of failures in system
- n Are these assumptions feasible or practical for a large-scale distributed system?
 - n Answer would seem to be no



Privacy

- n Privacy is **freedom from unauthorized intrusion** (M-W)
- n In physical world, privacy is easy to define and maintain
 - n “Close the door”, “Send letter in envelope”, ...
- n What about the digital world?
 - n What kind of privacy is “reasonable” to expect?
 - n What kind of privacy corresponds to the “classical” privacy?
- n Encryption can be used to protect personal data
- n What about personal information stored by others?
 - n Store needs to keep customer registry to function
 - n How should that information be kept and protected?



Anonymity

- n Anonymity seen as a way to protect privacy
- n Pseudonyms (e.g., user-picked ID) provides a simple form of protection
- n But pseudonyms are not enough
 - n Record company knows IP address
 - n IP address reveals ISP
 - n ISP has logs to tell who used the IP address
 - n Lawsuit follows
- n Pseudonyms also allow for user tracking
- n How to provide true anonymity on a P2P network?
- n Several solutions: FreeNet, **Achord**, Tarzan, Herbivore



Achord: Basics

- n Achord is a censorship resistant Chord
 - n Note: Censorship resistance not quite same as anonymity
- n Analysis about which Chord functionality is vulnerable to revealing the identities of nodes
- n Chord (or any DHT) is suitable for storage networks
 - n Guarantees that data will be found
 - n Bounds on the number of messages needed
- n Other anonymous networks (e.g., FreeNet) have no guarantees
 - n In FreeNet, less popular data may disappear
 - n No guarantees about finding any content
 - n No guarantees about number of messages
 - n But FreeNet provides more anonymity than Achord



Key Properties of Censorship Resistance

1. Possible to insert data without revealing the identity of the inserter
 - n Cannot censor by attacking those who insert information
 2. Possible to retrieve data without revealing the identity of the retriever
 - n Cannot censor by attacking those who want information
 3. Difficult to introduce a new node such that it will be responsible for a given document
 - n Cannot censor by deleting documents
 4. Difficult to identify node which is responsible for a given document
 - n Cannot censor by attacking the responsible node
- n (Especially) last point not fulfilled by Chord
- n Chord returns address of responsible node
 - n Problem with implementation, not a fundamental weakness



Achord and Chord

- n Node identity is SHA-1 hash of IP address
 - n Virtual nodes numbered and hashed
 - n Fulfills property 3
- n Each node knows $O(\log N)$ other nodes (finger table)
 - n Achord attempts to limit knowledge to this
 - n Attempts to fulfill property 4
- n Finding successor is Chord's fundamental operation
 - n Iterative and recursive methods
 - n *Find_successor* lets node find out what keys other node is responsible for
 - n Achord never returns *find_successor* to requesting node
 - n Achord maps keys to values
 - Chord maps keys to nodes



Achord: Finding Successor

- n No *find_successor* returned in Achord
 - n *Find_successor* is used, but the actual successor is not revealed to the requesting node
- n Instead, *connect_to_successor*
 - n Value is tunneled back to the requesting node
 - n Same for inserting a value
- n Provides anonymity
 - n Tunnel node cannot know who is requesting
 - Could be immediate requester or someone else
 - n Identity of the node storing a key is not shown
- n Above takes care of retrieving and inserting keys
- n Overlay maintenance requires new procedures



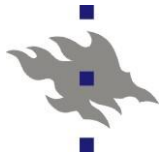
Overlay Maintenance

- n Recall: To join, new node must find its successor
 - n Call *find_successor* with own ID
- n Achord restricts use of successor and predecessor
 - n Only needed in a few cases, easily identified
- n Node n calls *find_successor(n)* to join network
 - n Benign call, anyone can verify that this is OK (needs IP address)
 - n In fact, a node **must** know its successor
- n **Rule 1:** Only node with ID n is allowed to call *find_successor(n)*
 - n Implies recursive processing of join is not possible
- n **Rule 2:** Only iterative processing of *find_successor* possible
 - n $O(\log N)$ nodes learn about a new node



Predecessors

- n Node needs to access predecessor field on other nodes in a single case
 - n Periodic stabilization and ring maintenance
- n Possible to determine if access to predecessor field is valid
- n If n' is successor of node n , then:
 - n n has called *find_successor*(n) which ended up at n'
 - n n' sets predecessor to n
 - n n' keeps list of predecessors, only most recent can access it
- n **Rule 3:** A node can access predecessor field on another node only if it was previously the predecessor and has not accessed the field since the value changed



Finger Tables

- Achord replaces Chord's finger table maintenance
 - Chord calls *find_successor* for each finger table entry
- Node updates its finger tables by picking a random node n' from its current finger table
 - Call $n'.find_best_match(i)$, where i is index to n' 's finger table
 - n' knows IP of n , can calculate the best match for n 's finger table slot i^{th} position
- **Rule 4:** Finger tables updated with *find_best_match* which returns a new IP address only if that node is a better match than the current node
- Nodes can collect IP addresses of others
 - Can get $O(k \log N)$ addresses



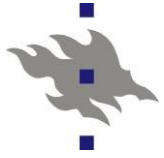
Achord: Issues

- ⌈ Possible to attack Achord if you have access to a large number of IP addresses
 - ⌈ Higher probability to be responsible for a given document
 - ⌈ Must limit number of virtual nodes?
- ⌈ Achord maybe not as anonymous as FreeNet
 - ⌈ Key and node IDs can be used to guess if a node sent a message
- ⌈ Nodes can learn about others during stabilization
 - ⌈ Extent is still unclear



Achord: Summary

- n Achord adds censorship resistance to Chord
- n 4 basic properties of censorship resistant systems
- n Basic idea:
 - n Provide anonymity
 - n Limit a node's knowledge about other nodes
- n Hard to provide total anonymity and good performance
 - n Tradeoff between the two
 - n Need more investigation
- n What is required from an anonymous system?
- n What is acceptable performance?



P2P and Copyright

- n What did Napster do wrong?
 - n First lawsuits against Napster after only a few months
 - n Eventually, Napster had to shut down
- n Reason for lawsuits: Copyright violations
 - n Users on Napster were sharing files without permission
 - n Copyright holders (= record companies) have the right to protect their rights
- n What can we learn from this case?
 - n Especially from the point of view of P2P software developer
 - n How should you build your system?
 - n What kinds of mechanisms can you use to avoid liability?
- n Recent rulings have gone against file sharing
 - n Most networks being shut down



What is Copyright?

n Copyright is:

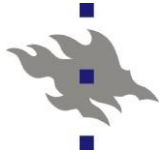
“A form of intellectual property that grants its holder the legal right to restrict the copying and use of an original, creative expression for a defined period of time.”

n Copyright holder has **exclusive** rights to:

- n Make and sell copies of the work (including electronic copies)
- n Import or export the work
- n Make derivative works
- n Publicly perform the work
- n *Sell or assign the rights to others (e.g., artist to record company)*

n **Only** the copyright holder can do these things

- n Everyone else is prohibited from doing them



Copyright and File Sharing

- n Copyright applies also to file sharing
- 1. Digital file is fixed
 - n Files being shared qualify as copyrighted works
- 2. Transmission of a file is reproduction
 - n Only copyright holder can reproduce the work
- n Any unauthorized reproduction of a copyrighted work is possibly copyright infringement

- n Our discussion concerns the Napster case and American copyright law
 - n European law similar, but varies from country to country
 - n New EU directives about copyright enforcement



Direct Infringement

- n Direct infringer is someone who is directly violating copyright law
 - n User who shares an unauthorized file
- n Direct infringer can be sued
 - n Record companies have sued many individual users who were sharing large number of files
- n In modern P2P file sharing networks, the presence of direct infringers is “guaranteed”
- n File sharing network would need to implement special mechanisms to prevent unauthorized sharing
- n Direct infringement does not (directly) concern the P2P software developer



What About The Developer?

- n Software developer not (usually) involved in creation or transmission of unauthorized copies
 - n Easy to avoid this in a P2P system
- n Copyright law can hold you accountable for the actions of others
 - n Also applies to other areas of law

Two kinds of secondary liability:

1. Contributory
2. Vicarious



Contributory Infringement

n *“One who, with knowledge of infringing activity, contributes to the infringing may be held liable.”*

Copyright owner must prove:

1. Direct infringement

n Direct infringement must have happened by someone

2. Knowledge

n Accused knew of infringement

n Actually, “should have known” is enough

n Must have specific knowledge, “system is capable of infringement” is not enough

3. Material contribution

n Accused must have contributed

n Providing “site and facilities” (e.g., search) is enough



Vicarious Infringement

- n Employer is responsible for actions of employees
 - n Right and ability to supervise and financial benefit

Copyright holder must prove:

1. Direct infringement

2. Right and ability to control

- n Must show that accused has right and ability to control the direct infringement

n *Napster: Ability to block user accounts is control*

3. Direct financial benefit

- n Accused must get direct financial benefit from infringement

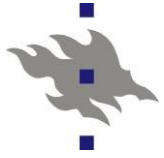
n Actually: “direct” and “financial” not important, any benefit is enough

n *Napster: Infringing material brings more users, makes company more attractive to investors*



Vicarious Infringement: Note

- n Vicarious infringement has no requirement of knowledge
- n Possible to be completely unaware of infringing activity and still be liable
- n Strong incentive to monitor your users
 - n If you do not monitor, you take a big risk



Possible Defenses

n No direct infringement

- n No direct infringement, no indirect liability
- n Hard to prove in a P2P file sharing network

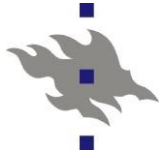
n Betamax defense: “Capable of substantial non-infringing uses”

- n Originally from Sony Betamax VCR case
 - Device capable of “substantial non-infringing uses”
 - No indirect liability
 - Actual use does not matter, “capability” is enough
- n *Napster: Betamax does not apply to vicarious infringement*
- n *Napster: Betamax defense applies only until you are notified of infringement*



More On Betamax Defense

- n Recent interpretations have two implications
 1. Betamax does not apply to vicarious liability
 - n Control and benefit are dangerous
 - n “Service” or “community-building” models are dangerous
 - These usually include some form of control
 2. When you are notified, you must do “something”
 - n What is “something”?
 - n *Napster: “Something” may be limited by the P2P technology*
 - In a fully decentralized network, not possible to do much
 - n Copyright owners argue designers should design for this case
 - This point not accepted by courts
- n Extent and applicability of Betamax defense still unclear



One More Defense

- n DMCA Section 512 “Safe Harbors”
 - n Similar new copyright directives in Europe too
- n Only apply to “online service providers” if infringement involves any of:
 - n Transitory network transmission
 - n Certain kinds of caching
 - n Storage for others (e.g., web hosting)
 - n Information location tools (e.g, search engine)
- n Safe harbors very tightly defined
 - n Consult a lawyer
- n This defense (also) failed for Napster



Lessons and Guidelines

n Make and store no copies

- n Even a copy in RAM can be considered a copy!
- n Creating copies makes you a *direct infringer*
- n Not really a problem for P2P developer (except caching?)

n Total control or total anarchy

- n Contributory infringement: Knowledge and contribution
 - Hard to avoid contribution (software is contribution)
 - When you “know”, you must “do something”
 - “Something” depends on architecture
 - **Either full control over users or no possibility to do anything**
- n Vicarious infringement: Control and benefit
 - Again, benefit hard to avoid (defined very loosely)
 - What is “control”?
 - **Either monitor users or make monitoring impossible**



Lessons and Guidelines

n Sell software, not services

- n Vicarious liability maybe biggest threat to P2P developer
- n Service model usually has possibility for “control”
- n Stand-alone software is out of developer’s control
 - For example, VCR manufacturer has no control over users
 - Remember: No automatic updates, etc.

n Can you deny knowledge about user activities?

- n Contributory liability depends on knowledge
- n Can you plausibly deny knowledge?
 - Rememeber: “Should have known” may be enough!
- n Don’t promote infringing uses
 - May mean no customer support
- n Again, total control or total anarchy



Lessons and Guidelines

n What are your “substantial, non-infringing uses”?

- n P2P systems very general purpose, don't think too small

n Don't promote infringing uses

- n No screen shots with Beatles songs in marketing material :-)

n Disaggregate functions

- n P2P system needs several components: search, management, ...
- n Split them over several entities (companies)
- n Responsibility of each entity limited to what it controls
- n Some entities may be better protected
 - For example, search entity may fall under DMCA safe harbor

n Don't make money out of infringing activities



Lessons and Guidelines

n Give up end-user license agreement (EULA)

- n EULA is a contract, may imply control

n No “auto-updates”

- n Auto-updates are “control over users”

n No customer support

- n Present no evidence that you have helped a direct infringer
- n Even reading a message from customer may be “knowledge”
 - For example, user asking about problems downloading “Matrix”

n Be open source

- n Hard to show “control” or “financial benefit”
- n But: “Benefit” defined very loosely by courts
- n But: If “dangerous” parts are open source, you can build business on safer ground (additional services)?



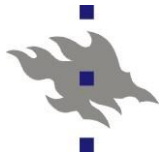
Future of File Sharing

- n What does future look like for file sharing?
- n Record companies going after individual users (i.e., the direct infringers)
 - n Even got a conviction (Jammie Thomas)
- n BitTorrent communities shut down
 - n Sites with links to illegal content
- n Illegal file sharing will not go completely away
 - n May degrade into an underground activity
- n Legal alternatives will become more popular?
 - n Buying digital content online



Pollution in File Sharing

- n A “pollution company” creates fake files
 - n Files appear to be “legitimate” (read: popular songs)
- n File contents are not what the metadata says they are
- n Searching is only based on metadata
 - n Users will get bad files instead of good files
 - n Bad files spread through the system
- n **Two intended outcomes:**
 - n More bad copies than good copies
 - n Users get frustrated and stop using the system
- n One such “pollution company” is Overpeer



Types of Pollution

n Content pollution

- n Correct metadata, but content is “modified”
 - For example, insert white noise in the middle of a song

n Metadata pollution

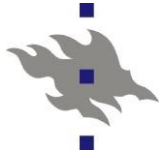
- n Metadata does not match the content (but content might be ok)

n Intentional pollution

- n Pollution is done on purpose

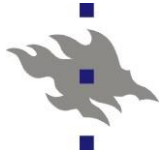
n Unintentional pollution

- n Accidental pollution, e.g., truncate song while ripping, typo in metadata, ...



How Much Pollution is There?

- n Experiment with several popular songs
- n Types of pollution found:
 - n Files un-decodable, songs too short or long, modified content
- n Result: Pollution is extremely wide-spread
- n Up to 70% of copies of some songs were polluted
 - n Percentage of polluted copies higher for popular songs
- n Simple rating schemes are not enough
 - n Even if one bad version is “rated out”, new polluted versions appear too fast



Anti-Pollution Techniques

n Detection with downloading

- n Download all or part of file to determine pollution
- n Match file contents to a well-known trusted source
 - For example, hash contents
- n Users filter out bad copies
 - User downloads file, but does not share bad copies
 - Need incentives?

n Detection without downloading

- n Detect polluted copies without downloading any part of file
- n Download files only from people you trust
- n Web of trust: Same idea, extended
- n Reputation systems



Online Music Stores

- n Answer from record companies to file sharing
 - n Nothing to do with P2P as such, but a competing technology
- n First was Apple's iTunes Music Store (iTunes)
- n Many others followed:
 - n Napster 2, Walmart, Musicload.de, ...
- n Idea behind online music stores:
 - n Users pay a small amount for a music file (with DRM)
 - File downloaded from store to user's computer
 - n Can also buy complete albums
 - n Can play songs on computer or portable player, or burn to CD
 - n Price typically ~1 euro per song or ~10 euros per album
- n Goal: Provide experience similar to buying a real CD



Online Music Stores: User Rights

- n What user is allowed to do with music?
 - n How does it compare with buying a traditional CD?
- n With iTunes, you can do the following:
 - n Play song on 5 computers
 - n Transfer song to an iPod
 - n Burn song to a CD up to 7 times
 - n Share song with 5 computers on same subnet (e.g., home)
 - n Share song wirelessly to speakers
- n Digital Rights Management stops when burning a CD
 - n Can later rip to a music file without DRM (loss of quality)
- n Are you buying the song or a license?



Online Music Stores: Future

- n Currently iTunes and others very popular
- n In other words: **People are willing to pay for content**
 - n At least as long as it's a well-marketed and useful service
 - n Is this the best business model?
- n Trend towards payable media
 - n iTunes now sells/rents TV shows and movies
 - n DSL operators offer movies
- n Still long way from payable Internet
 - n Likely to happen in future
 - n Basic services will be free, have to pay for others
 - n Well-understood by people (e.g., cable or satellite TV)
 - n But needs much, much more work to work on Internet?



Chapter Summary

- n Security issues in DHTs
- n Privacy and anonymity
- n Napster legal case and copyright
- n Pollution in file sharing
- n Online music stores