Network Services

Dr. Manfred Hauswirth
Laboratoire de Systèmes d'Information Répartis
École Polytechnique Fédérale de Lausanne (EPFL)
http://lsirpeople.epfl.ch/hauswirth/

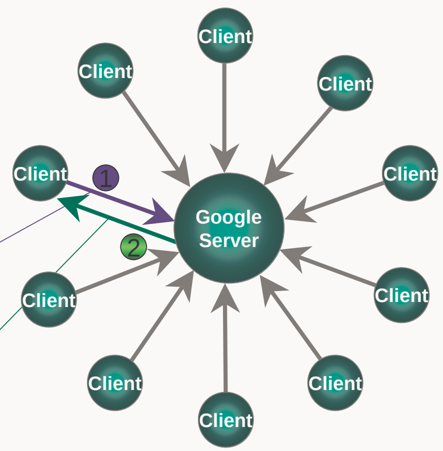
Oct 25, 2002 – Lecture 5

Overview

- Information system architectures
- Scalable Data Access Structures (SDAS)
- More Peer-to-peer systems: P-Grid
  - Searching
  - Building up a P-Grid
  - Request load balancing
  - Trust
  - Semantic gossiping
- Security
  - Certificates, Public Key Infrastructures (PKIs)
  - Firewalls, copyright, data protection, privacy
  - SSL, PGP
- E-Commerce: Macropayment vs. micropayment, SET, Millicent
- *Push systems (if time is available)*
  - *Concepts, components, communication model*

Centralized Information Systems

- Web search engine
  - Global scale application
- Example: Google
  - 150 Mio searches/day
  - 1-2 Terabytes of data (April 2001)



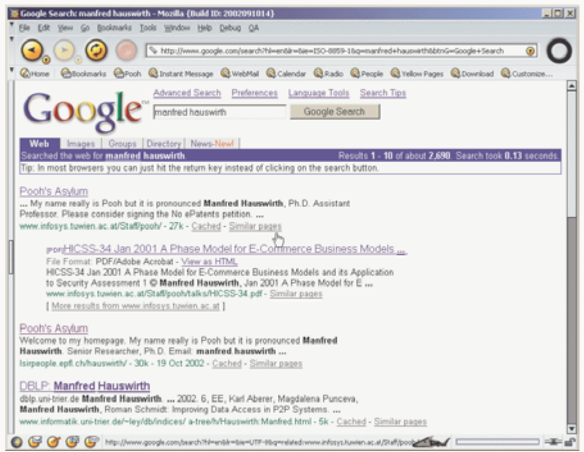
Find "manfred hauswirth"

Result: homepage of Manfred Hauswirth…

Google: 15000 servers

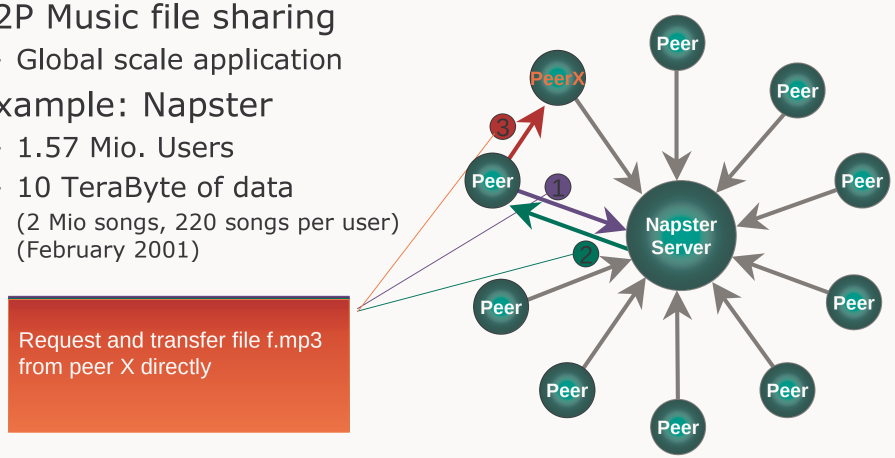
Google Assessment

- Strengths
  - Global ranking
  - Fast response time
- Weaknesses
  - Infrastructure, administration, cost
  - A new company for every global application ?



(Semi-)Decentralized Information Systems

- P2P Music file sharing
  - Global scale application
- Example: Napster
  - 1.57 Mio. Users
  - 10 TeraByte of data
    (2 Mio songs, 220 songs per user)
    (February 2001)



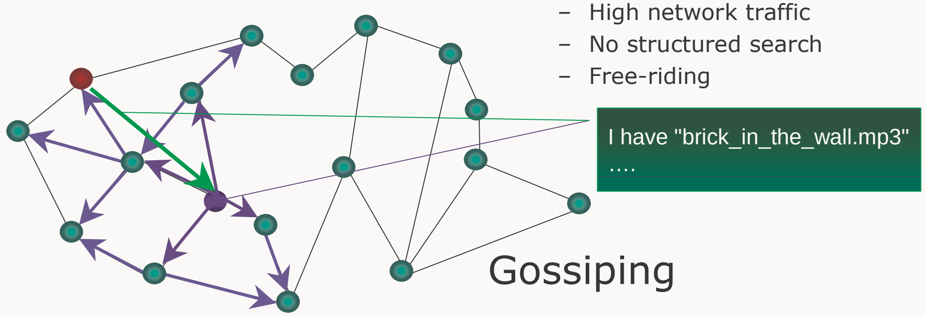
Napster: 100 servers

Lessons Learned from Napster

- Strengths
  - global information system without huge investment
    - exploit unused resources at nodes (space)
    - exploit users knowledge at nodes (annotation, e.g., annotate music files)
  - decentralization of cost and administration
  - set up a very large scale information system without heavy investment (as e.g., Google)
  - keeping content where it is created
- Weaknesses
  - business model: copyrighted material
  - server is single point of failure
  - therefore it can, for example, be shut down

Fully Decentralized Information Systems

- P2P file sharing
  - Global scale application
- Example: Gnutella
  - 40.000 nodes, 3 Mio files
    (August 2000)
- Strengths
  - Good response time
  - No infrastructure
  - Fully decentralized
- Weaknesses
  - High network traffic
  - No structured search
  - Free-riding



Gnutella: no servers

Napster vs. Gnutella

		Napster	Gnutella
Resources	search	**central**	**decentral**
	file exchange	**decentral**	**decentral**
Knowledge	schema	**central**	**trivial**
	annotation	**decentral**	**decentral**
		Partially decentralized	Self-Organizing

Decentralization – Self-Organization

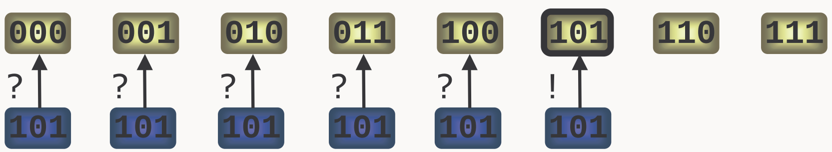
- Decentralization
  - strategy to avoid performance bottlenecks (scalability), single points of failure, points for legal attack
  - no central coordination, no central database, no peer has a global view of the system
- Self-organization
  - global behavior emerges from local interactions
  - cooperation/coordination without central control
  - Decisions based on local (or missing) information (autonomy or non-determinism)
- P2P: Towards symmetric system architectures with some desired (or observed) global behavior

Self-Organization and Efficiency

- Self-organization
  - Can be costly if done wrong
- Example: Search Efficiency in Gnutella
  - Search requests are broadcasted
  - Anectode: the founder of Napster computed that a single Gnutella search request (18 Bytes) on a Napster community would generate 90 Mbytes of data transfers

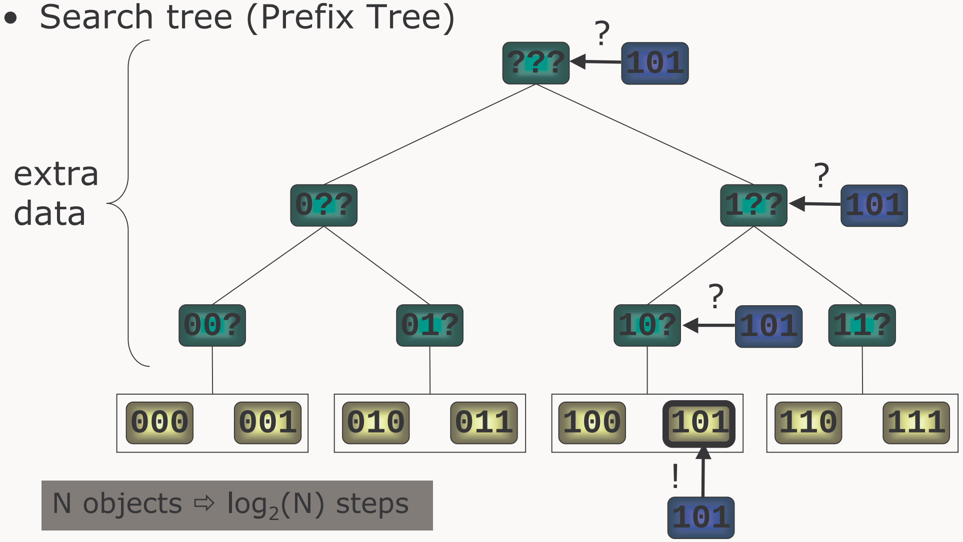
Data Access Structures

- Given: a search request (e.g., a name)
- Find all data objects that correspond to this request quickly (e.g., having the name)
- Sequential search does not scale
  - N objects: N steps
- Sequential search does not scale



Data Access Structures

- Search tree (Prefix Tree)



Tree-based Data Search

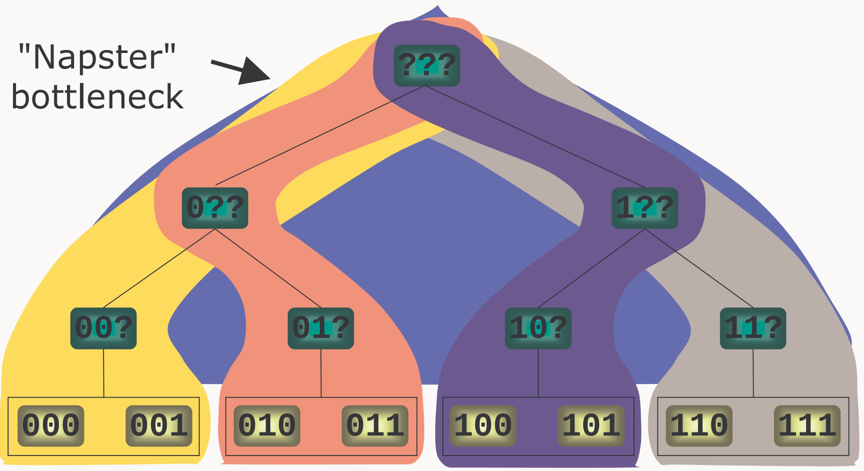
- For N data objects
  - Sequential search requires on average N/2 steps
  - Tree search requires $\log_2(N)$ steps

data objects	sequential	tree
1.000	500	10
1.000.000	500.000	20
1.000.000.000	500.000.000	30

Scalable Data Access Structures - 1

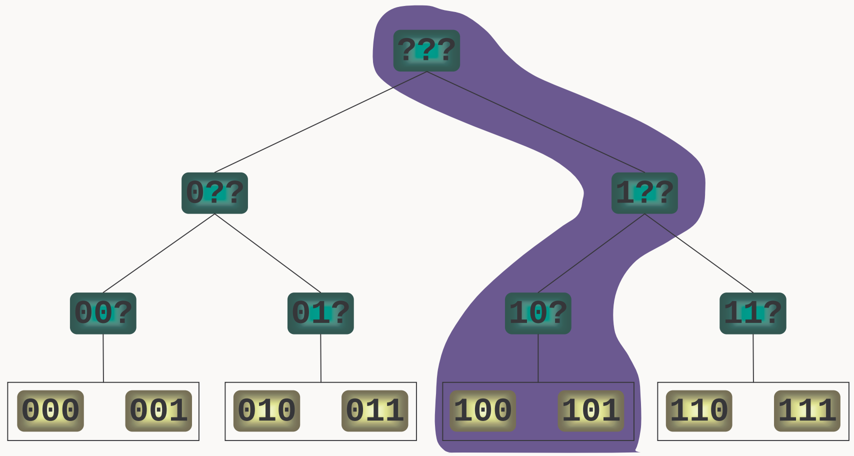
- Assume number of data objects >> storage of one node
  - Distributed storage
- Given a data access structure
  - Size of data access structure = number of data objects
  - Therefore: Size of data access structure >> storage of one node
- Problem: where to store ?

Scalable Data Access Structures - 2

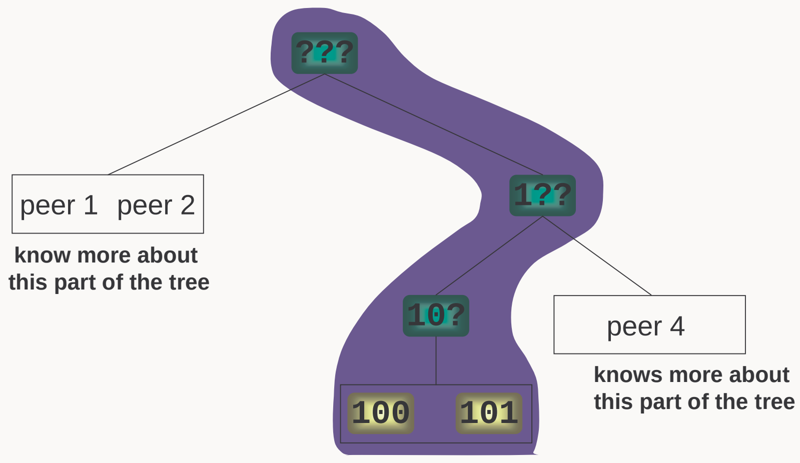


Scalable Data Access Structures - 3

- Associate each peer with a complete path

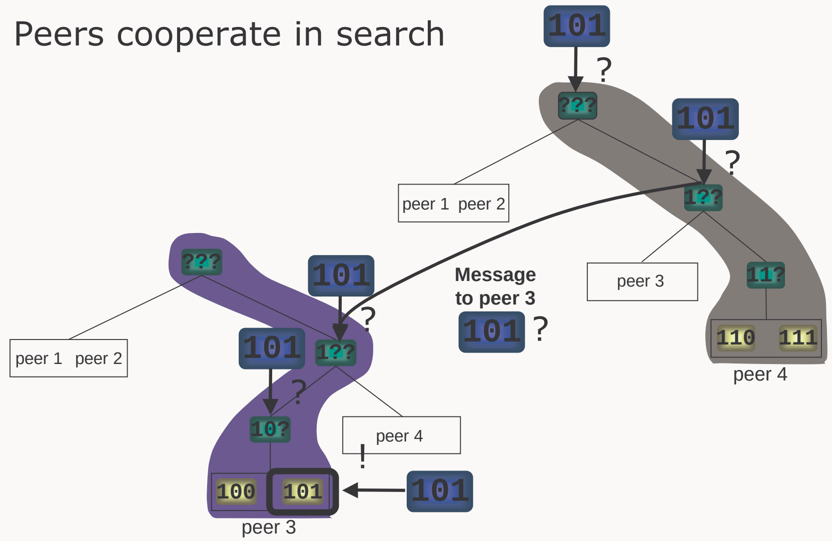


Scalable Data Access Structures - 4

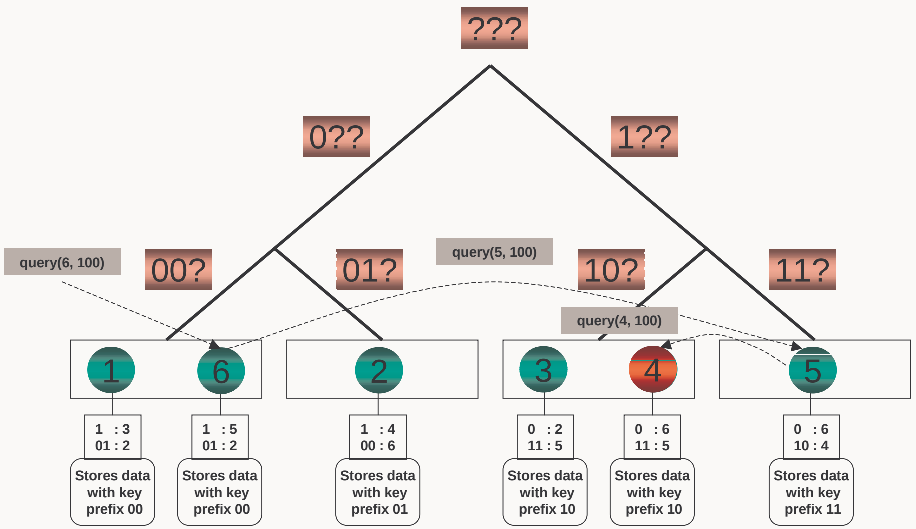


Result: P-Grid

- Peers cooperate in search



P-Grid Queries



SDAS Discussion

- Scalable Data Access Structures
  - Require only $Log_2(N)$ storage at one node
  - Support $Log_2(N)$ search
  - Are therefore scalable in N (beyond $N=10^{10}$ as in Google)
- Idea found in
  - OceanStore
  - DNS
  - Parallel and distributed DBMS

Construction of SDAS?

- Standard methods
  - Each node has an identifier, compute position in the tree from the identifier
  - Structure of tree depends on distribution of identifier. What's the connection ?
    - Nodes can no more choose which data they want to store and which requests they want to answer (autonomy)
  - Each node asks a coordinator for its position in the tree
    - Coordinator is the bottleneck

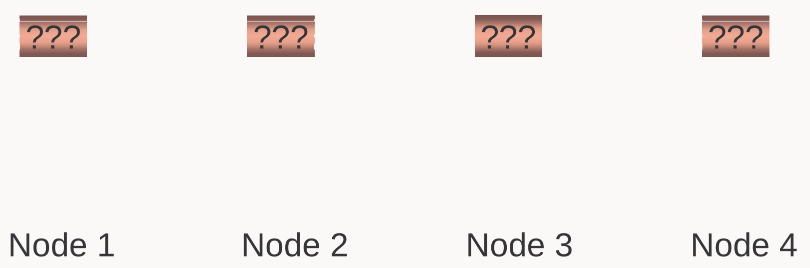
P-Grid Construction

- Idea
  - Replace the coordinator by a random process
  - P-Grid construction algorithm
  - distributed, decentralized, randomized

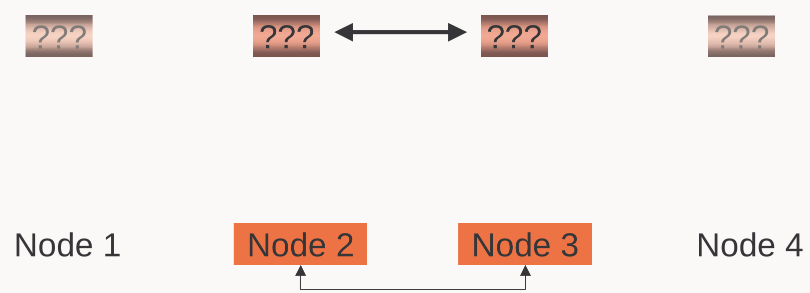
```
when two peers meet
   if a maximal path length is not reached
      try to extend "their" path in tree
      do the necessary bookkeeping
```

- Requires some care in order to work efficiently

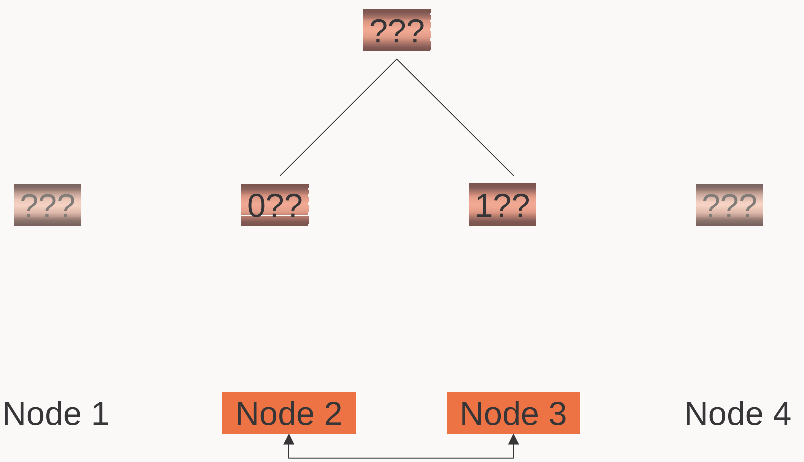
Random Meetings



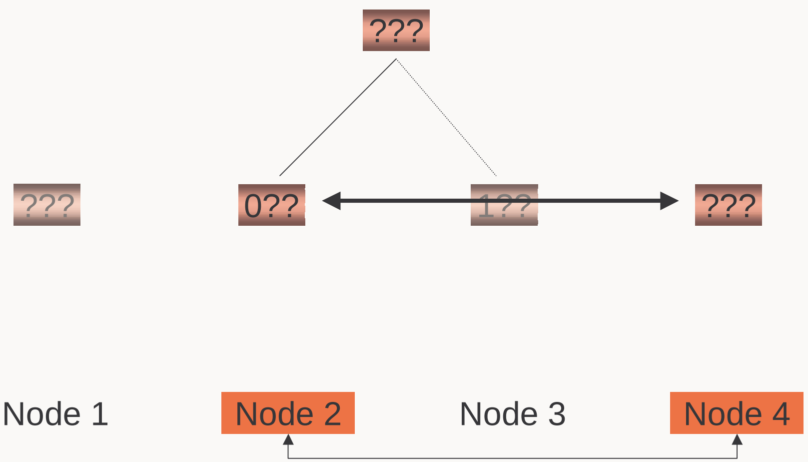
Random Meetings



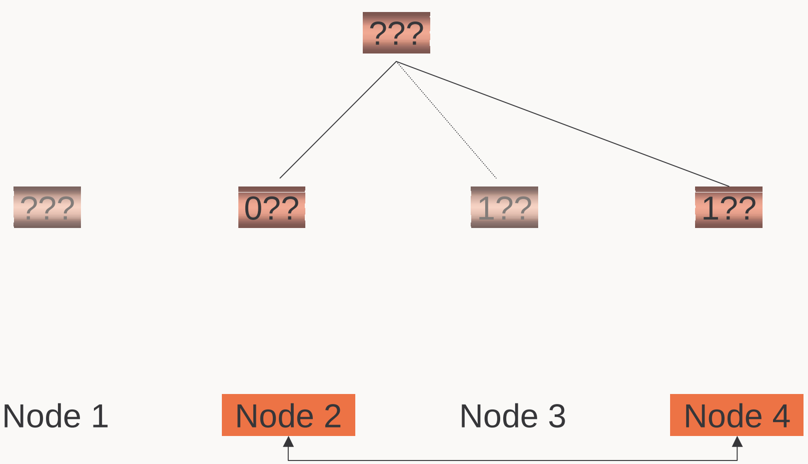
Random Meetings



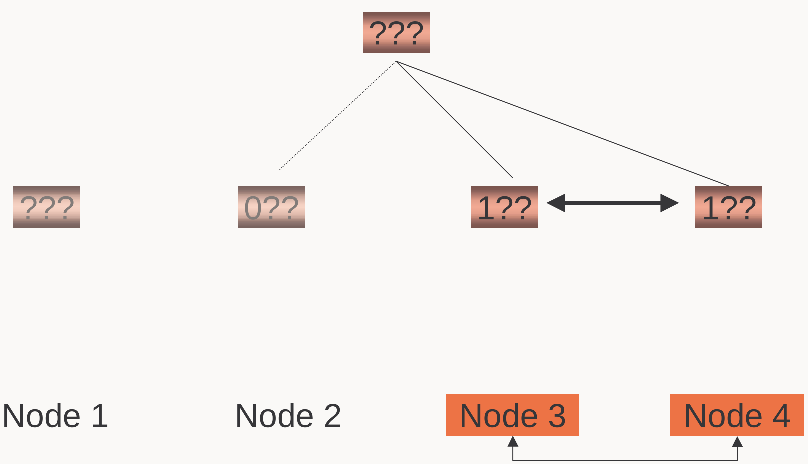
Random Meetings



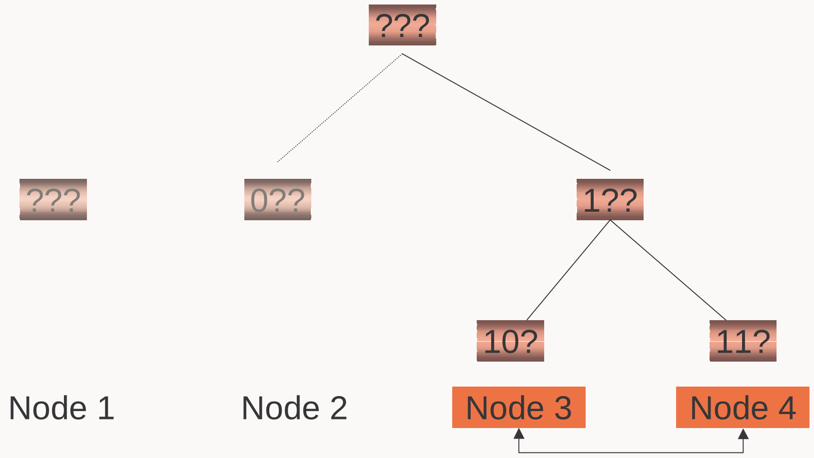
Random Meetings



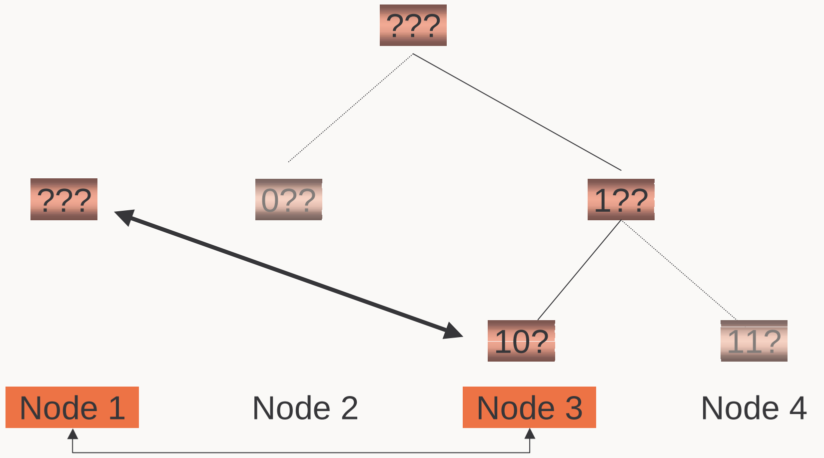
Random Meetings



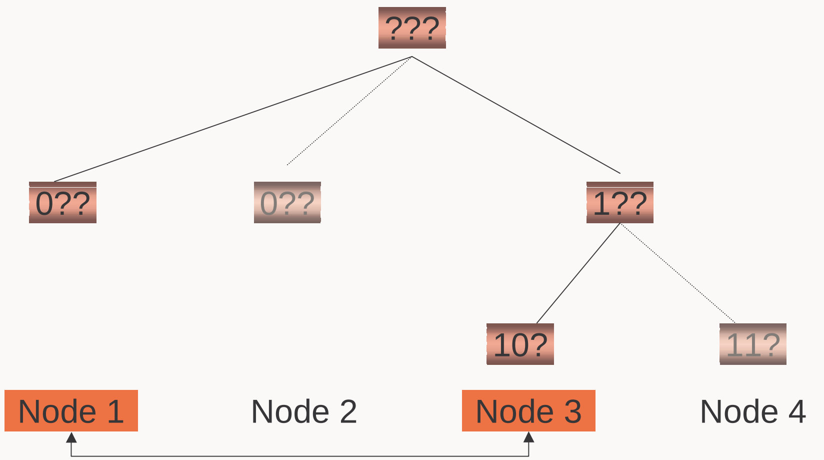
Random Meetings



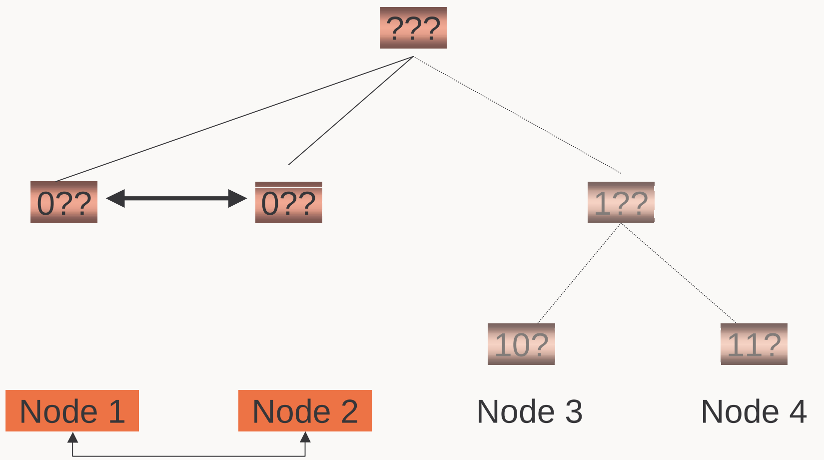
Random Meetings



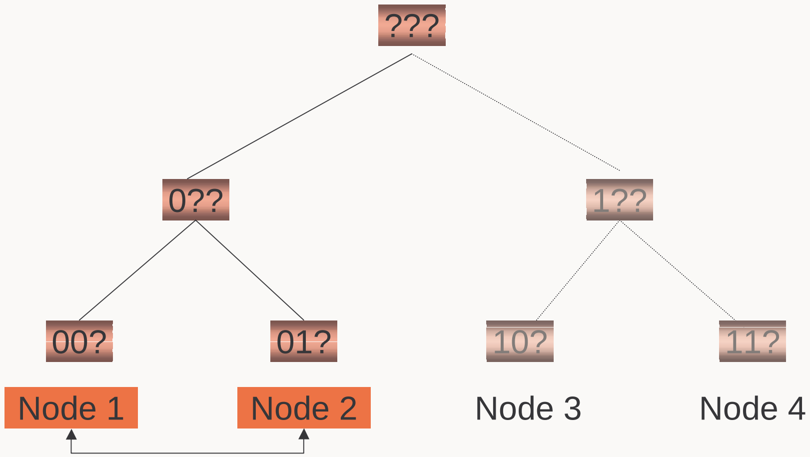
Random Meetings



Random Meetings

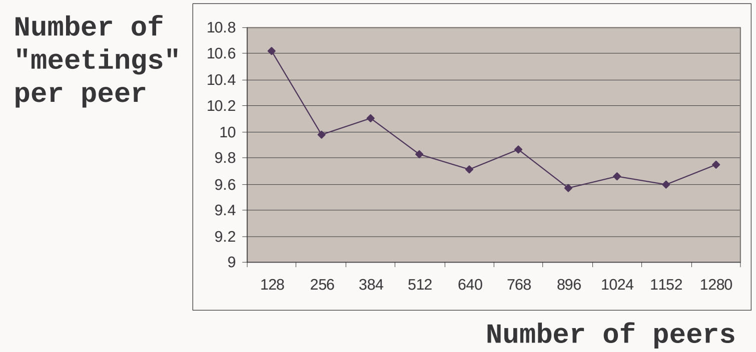


Random Meetings



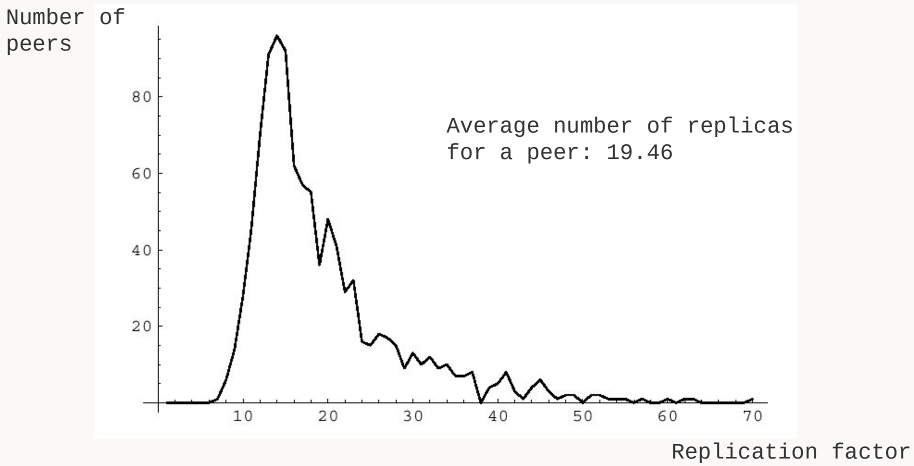
Efficiency of P-Grid Construction

- Constructing a tree of depth 6 (64 leaves)



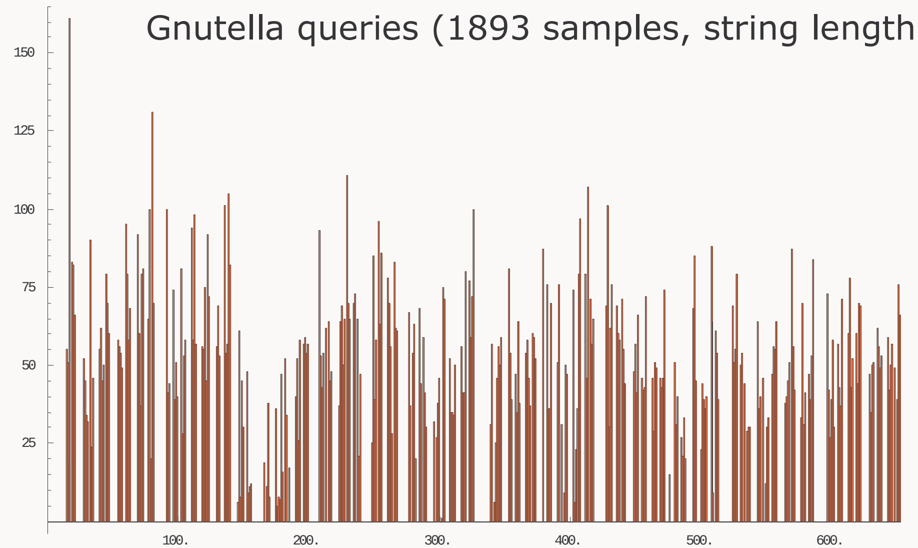
Replication?

(n = 20000, k = 10, refmax = 20)



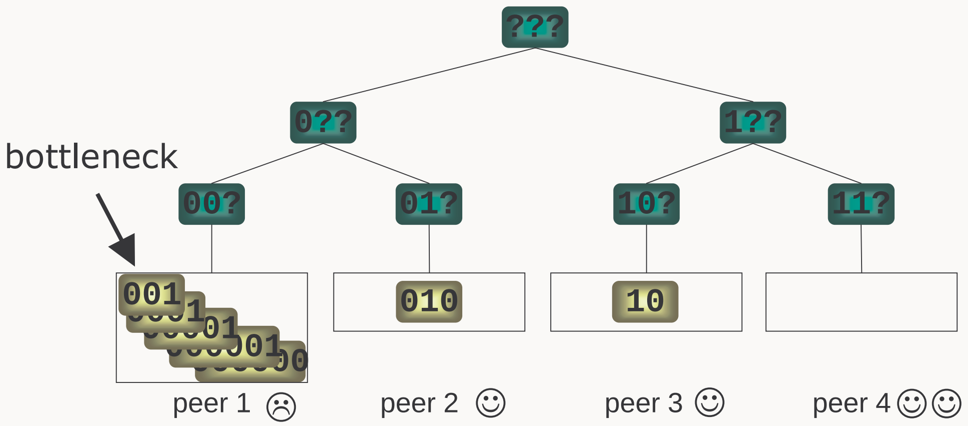
Non-uniform Data Distribution

Gnutella queries (1893 samples, string length 4)



Non-uniform Data Distribution

- Construct a tree of depth 2 for the following data:
  10, 01, 001, 0001, 00001, 000001, 000000



P-Grid Construction Balancing Storage Load

- It makes no sense to create leaves in the tree for data occuring rarely

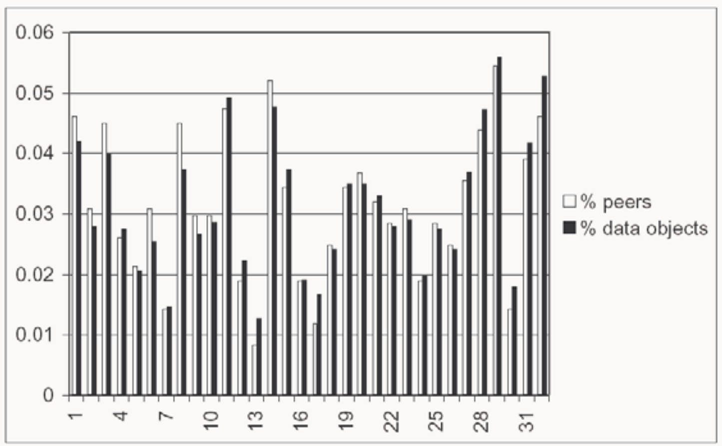
- Every peer stores initially some data

```
when two peers meet
      peer extends path only if #data items > ε
            do the necessary bookkeeping
      otherwise data exchange (duplicate generation)
```

- Requires some care in order to work efficiently and to correctly balance the storage load

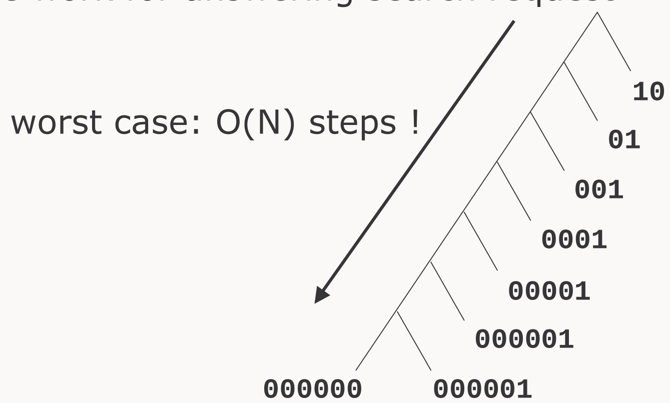
Result

… and still converges quickly



Unbalanced Search Trees

- Problem:
  tree will be deeper where more data items
  ⇒ more work for answering search request



Analysing Required Messages

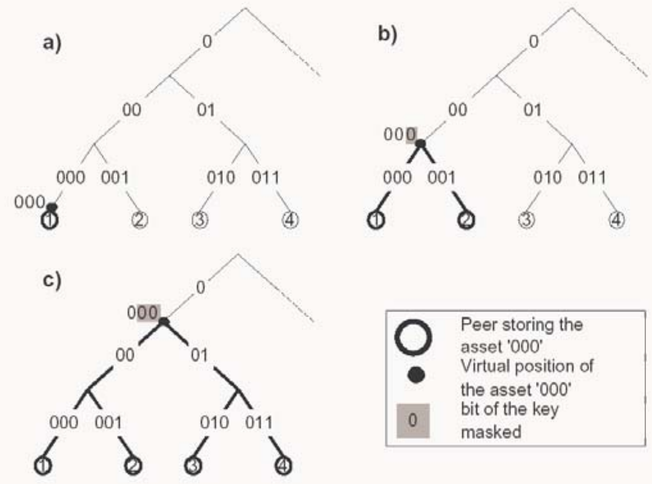
- Assume only the number of messages required for a search is relevant
  - Multiple nodes in the tree can be traversed without sending a message

Theorem

> IF the probability for a reference to another node occuring in the reference lists is equal for all references that possibly can occur,
> THEN the number of messages required for a search is $O(\log_2(N))$ no matter what shape the P-Grid (tree) is.

- Equal probability can be achieved by systematically merging the reference lists

Request Load Balancing – 1



Request Load Balancing – 2

- Gamma: popularity threshold
  - if access measure > gamma a new popularity level is granted
- Optimum: 0.2 (< 0.2: too much competition)

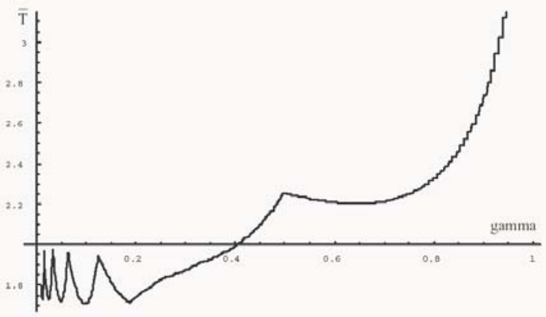


Figure 5: Average Response Time, Finite Capacity

Self-Organization in P-Grid

- Nodes decide through local agreement
  - on their position in the search tree when they meet
  - whether to deepen a search tree based on storage load
- Nodes balance data through local operations
  - Reference distribution
    - required for search efficiency
  - Replica distribution of data objects
    - required for search reliability
- Global "agreements" are only on
  - Type of search requests
  - P-Grid organisation

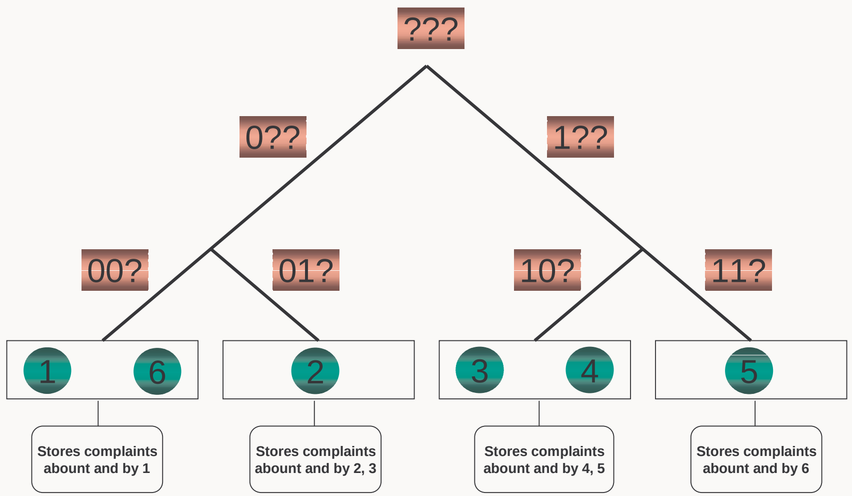
Practical Aspects of P-Grid

- Implementation exists: feasible
  [IEEE Internet Computing 2002]
- Analysis shows that indexing overhead is reasonable for typical setting
  [Coopis 2001]
- Algorithms for additional replication of more frequently requested data objects
  [ICME 2002]
- Update mechanism based on gossiping
  [EPFL-TR 2002]
- Identification and management of dynamic addresses
  [EPFL-TR 2002]
- Application for storing reputation data
  [CIKM 2001]
- More complex queries can be supported
  (regular expressions, paths, joins)

Trust Management based on Reputation

- Approach
  - Record complaints by peers
  - Build a decentralized data warehouse based on P-Grid
  - Compute average number of complaints
  - Retrieve from the data warehouse all complaints on (and by) a peer
  - Also assess the trustworthiness of the peers reporting theses numbers
  - Apply a weighting formula and decide
- Result
  - Even with a large fraction of cheaters (25% are cheating 25% of the time) they can be reliably recognized

Using P-Grid to store Trust Data



Updates in P-Grid

- Most P2P systems consider data to be read-only
- The goal is not to achieve complete consistency but rather to know what is the probability of a correct answer given certain model parameters
- Scenarios:
  - A query occurs during an update
  - A peer is online while an update is processed
  - A peer is offline while an update is processed
  - A peer crashes or fails
  - The communication with a peer is temporarily disrupted

Types of Updates in P-Grid

- New peer joins P-Grid: all the peer's data must be communicated to the responsible peers and their replicas
- New data item is inserted
- Existing data item is updated
- Due to a re-organisation in the P-Grid a new peer becomes responsible for a certain data item
- Management of dynamic IP addresses

P-Grid's Update Algorithm (Push Phase)

- *p* receives an update request *update(K, V, p_f, R_f)* with *K*: key of the data to update, *V*: new value, *p_f*: requesting peer, *R_f* replica list of requesting peer
- *p* propagates the update request to *R_p* \ *R_f*
- *p* sends *R_p* \ *R_f* to *f* so that *f* learns about new replicas
- *p* discovers additional replicas from *R_f* \ *R_p*, contacts them and updates its replica list: *R_p* = *R_p* ∪ *R_f*
- If *R_p* did not change for the last *L* update requests (or time period *T* or self-tuning parameter), then ascend the P-Grid search tree to find new replicas and contact them.
- If a peer *r_p* cannot be contacted then retry:
  - Check whether the ping counter has exceeded its maximum or the the maximum offline period of replica peers has expired.
  - If one of these 2 conditions is true remove *r_p* from *R_p*.
  - If *r_p* becomes online again and requests a state update of or other peers notify the peer that did the unsuccessful ping, then *r_p* is put into *R_p* again.

P-Grid's Update Algorithm (Pull Phase)

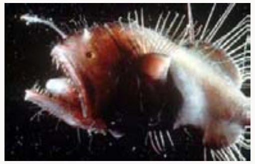
- Scenario 1: If a peer has been off-line it must try to get a consistent view of the data again.
  - *p* contacts some of its replicas randomly and asks for all updates for the time it had been offline.
- Scenario 2: A peer is online but communication is temporarily disrupted.
  - This is more complicated because the peer may not recognize that its communication with the rest of the P-Grid network is broken.
  - To get around: contact other peers (specifically its replicas) at regular intervals
    - Randomly ping a replica *r_p* from the list of active replicas *R_p*
    - Check whether the ping counter has exceeded its maximum or the the maximum offline period of replica peers has expired.
    - If one of these 2 conditions is true remove *r_p* from *R_p*.
    - If *r_p* becomes online again and requests a state update of or other peers notify the peer that did the unsuccessful ping, then *r_p* is put into *R_p* again.

Decentralization: Napster vs. Gnutella

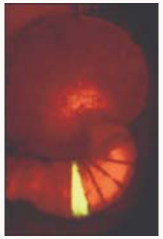
		Napster	Gnutella
Resources	search	**central**	**decentral**
	file exchange	**decentral**	**decentral**
Knowledge	schema	**central**	**trivial**
	annotation	**decentral**	**decentral**
		Partially decentralized	Self-Organizing

Why are Schemas Important ?

- Example: Searching biological databases
  - Without schema (like Google, Gnutella)



- Searching for data on "anglerfish"
  - Results will be precise
- This seems easy, but the same for "leech"
  - Organism leech
  - Authors: "Bleech", "Leechman", …
  - Protein sequences: …MNTS**LEECH**MPKGD…



Schema Heterogeneity

- Different databases – Different schemas
  - SwissProt: Find **<Species> leech </Species>**
  - EMBLChange: Find **<Organism> leech </Organism>**
- Standardization (global schema) ?
  - Music files: clear scope, simple semantics ☺
  - Scientific databases: different scope, distributed knowledge, little agreement, etc. ☹
- Hardest problem in information systems: semantic interoperability

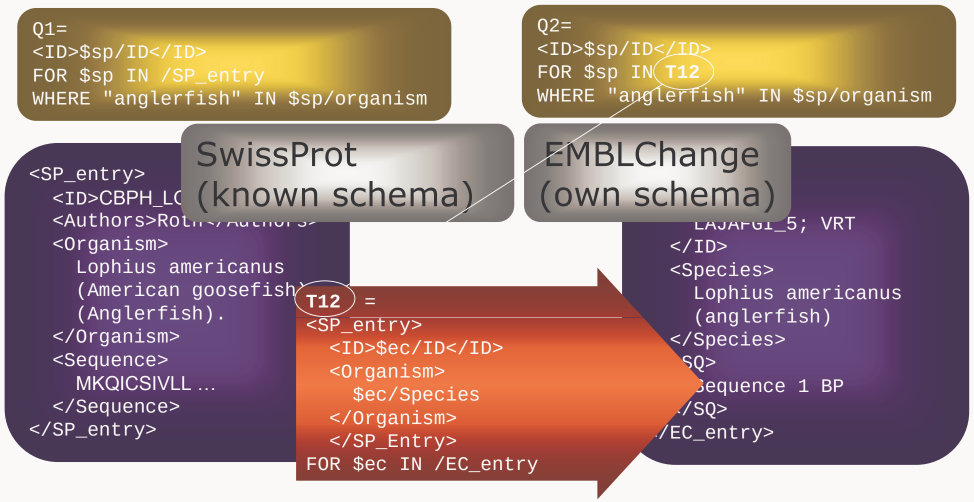
Translating Heterogeneous Schemas

- A non-expert may be able to relate
  - **<Organism>** ⇔ **<Species>**
  - **<Author>** ⇔ **<Authors>** etc.
- But what about
  - **<AaMutType>** ⇔ **<DnaMutType>**
  - **<FtKey>** ⇔ **<FtKey>**

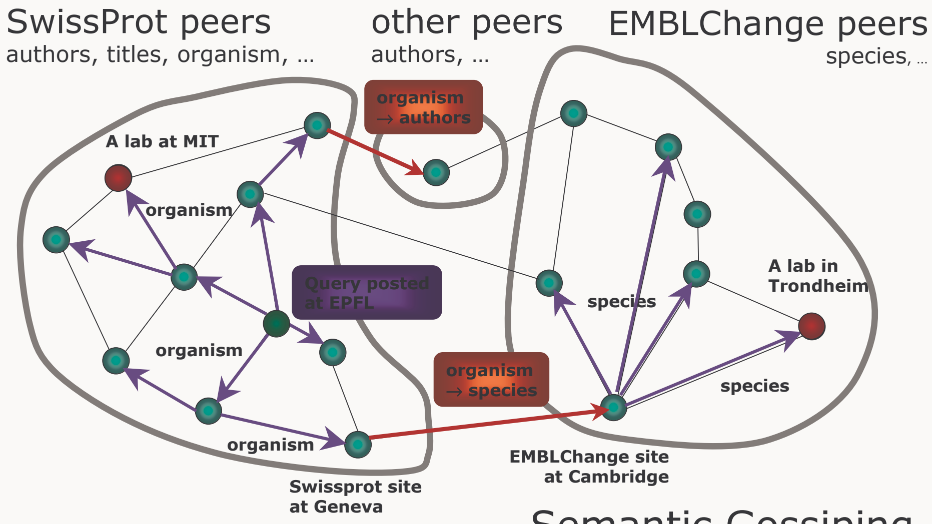
  in Swisschange and EMBLChange ?
- The answers can only be given by the experts … sometimes only by the data owners !

Approach: ask them to provide their translations from some "known" schema to their "own" schema (local step)

Local Semantic Interoperability (Translation)

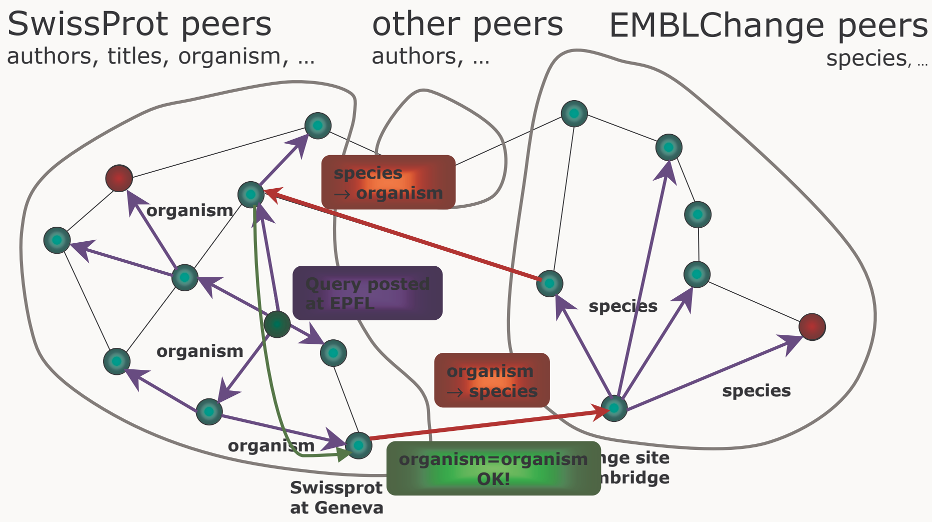


Global Semantic Interoperability



Semantic Gossiping

How to Detect a Semantic Agreement ?



Check what is preserved in cycles (semantic kernels) !

Semantic Kernels

- After a few translations the query returns
- T1(T2(T3(T4(Q))))
- In general: T1(T2(T3(T4(Q)))) != Q
- But there always exists Q' sucht that
- T1(T2(T3(T4(Q'(Q)))))= Q'(Q))
- Therefore an agreement exists on this query !

Research Questions

- Many fundamental problems
  - Erroneous agreements
  - Agreement on schema but not on data
  - Complex data types and mappings
  - Overlapping of data collections
- Approach: algorithms and tools
  - to automatically generate, detect and use local translations
  - identify which are correct with a high probability (via semantic kernels)
  - control of global search (via semantic gossiping)

Summary and comparison of P2P approaches

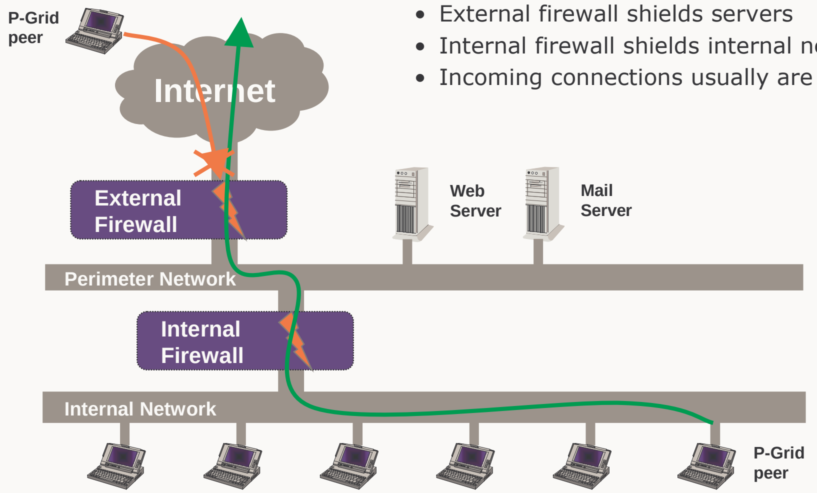
	Paradigm	Search Type	Search Cost (messages)	Autonomy
Gnutella	Breadth-first search on graph	String comparison	$2*\sum_{i=0}^{TTL} C*(C-1)^i$	very high
FreeNet	Depth-first search on graph	String comparison	O(Log n) ?	very high
Chord	Implicit binary search trees	Equality	O(Log n)	restricted
CAN	d-dimensional space	Equality	O(d n^(1/d))	high
P-Grid	Binary prefix trees	Prefix	O(Log n)	high

Dynamic IP Addresses/Mobility and P2P

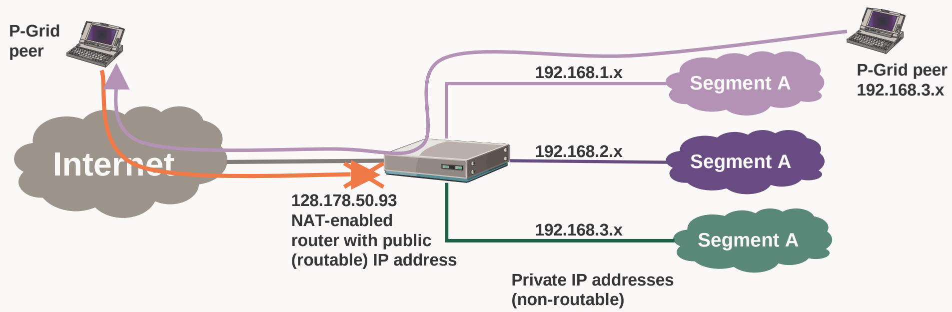
- Typically hosts have changing IP addresses
  - Dynamic Host Configuration Protocol (lease time)
  - Host mobility (physical mobility)
- No problem for pull-based P2P systems
  - New peer initiates a "permanent" connection to other peer(s) that route(s) requests to the new peer via this connection (for example, Gnutella).
  - No "permanent" connection ⇨ problem
- BIG problem for pushed-based P2P systems
  - Peers actively try to connect via a new connection (for example, P-Grid)
  - What if the IP address has changed in the meantime?
  - Location transparency? Migration transparency?

P2P problems: firewalls

- External firewall shields servers
- Internal firewall shields internal network
- Incoming connections usually are blocked

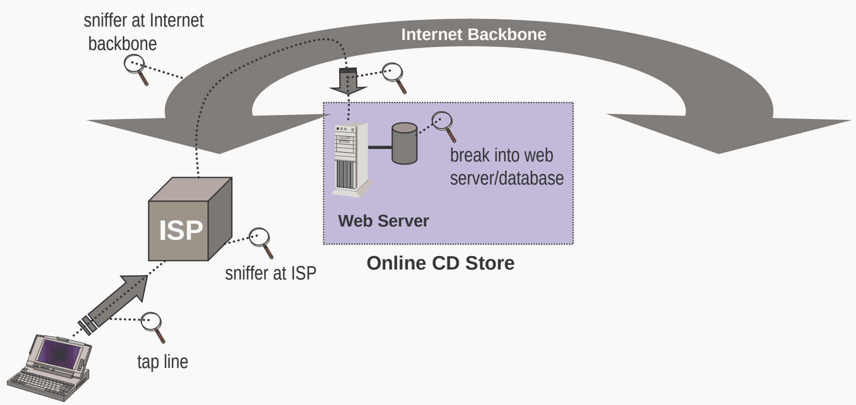


P2P problems: Network Address Translation



- NAT translates private (non-routable IP) addresses into public (routable) ones
- Unidirectional concept (from Intranets to Internet)
- Bi-directional possible, but difficult and thus usually not configured
- Many protocols are not NAT-friendly: VoIP, RTP, RTCP, IPSec, P-Grid, etc.

Life's not fair ...



Security - What it means

- Confidentiality
  - prevent unauthorized accesses
  - encryption
- Integrity
  - prevent unauthorized changes
  - message authentication codes (MACs)
- Availability
  - uninterrupted access
  - backup, prevent denial-of-service attacks
- Authenticity
  - prove origin of data
  - digital signatures

Security Threats

- Leakage
  - an unauthorized person tries to get hold of information belonging to or intended for somebody else
- Tampering
  - unauthorized alteration (including deletion) of information or programs
- Resource stealing
  - unauthorized use of facilities (such as memory, disk space, or network connections
- Antagonism
  - an interaction with a system takes place which does not result in a gain for the intruder but is annoying (vandalism)

Security Attacks (1/2)

- Eavesdropping / disclosure of information
  - an unauthorized intruder tries to read information which is sent over a network or is stored in memory
  - difficult to detect since they normally do not leave traces
- Masquerading
  - an intruder tries to use someone else's identity to gain access to the system
- Message tampering
  - unauthorized changes of network messages
- Replaying
  - network packages are stored and resent at a later time

Security Attacks (2/2)

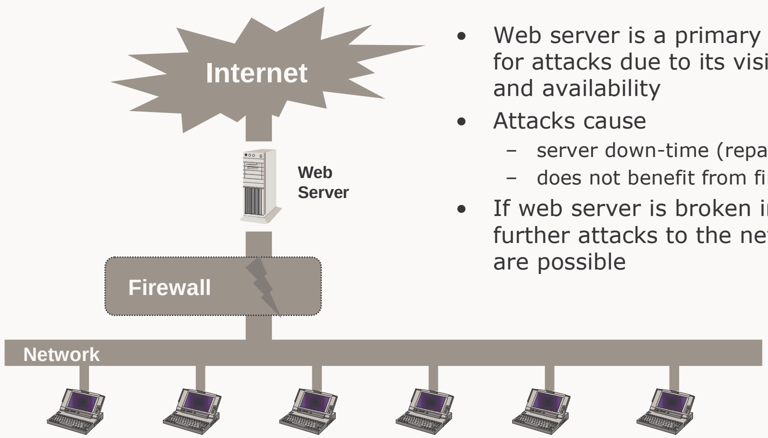
- Denial of service
  - DOS attacks make parts of a system unusable for other (legitimate) users by hogging, damaging, or destroying a resource
- Social engineering
  - an intruder gains access to a system by playing the role of someone else. He/She can try to convince a user to change his password to a given string or act as a system administrator and simply ask for the password for a special reason
- Exploits
  - use security holes in operating systems and software to gain access to a system
- Data driven
  - E.g., viruses and Trojan horses

Firewalls: Part of the Solution

- Isolates a network from the Internet
- Allows certain connections and blocks others
- **Firewall ≠ Security !!**
  - frequently a substitute for real problem fixing
  - many attacks by frustrated/dishonest employees
  - important but divert attention from real network problems, host vulnerabilities, poor planning, lack of organizational policies

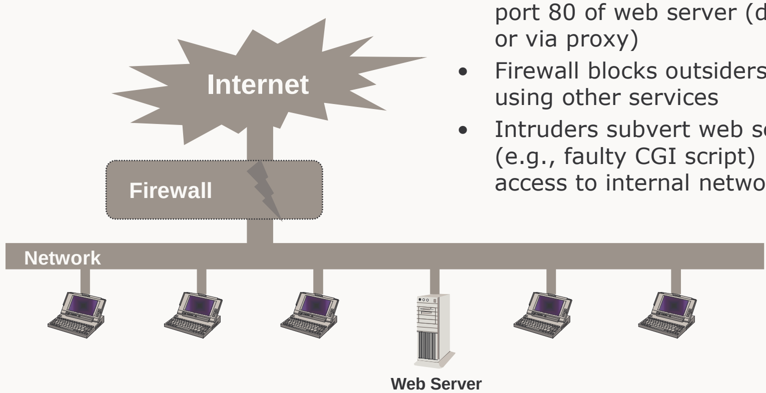
⇨ **Firewall = ADDITIONAL Security**

Firewall Placement (1/3)



- Web server is a primary target for attacks due to its visibility and availability
- Attacks cause
  - server down-time (repair)
  - does not benefit from firewall
- If web server is broken into no further attacks to the network are possible

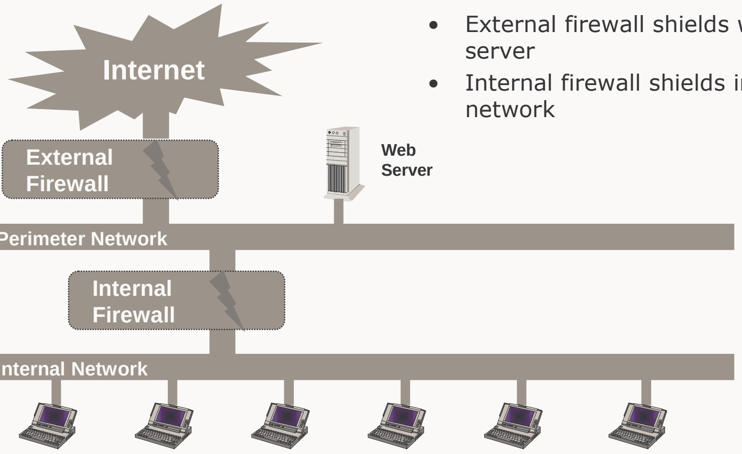
Firewall Placement (2/3)



- Firewall must allow access to port 80 of web server (directly or via proxy)
- Firewall blocks outsiders from using other services
- Intruders subvert web server (e.g., faulty CGI script) => full access to internal network

Firewall Placement (3/3)

- External firewall shields web server
- Internal firewall shields internal network



User Safety

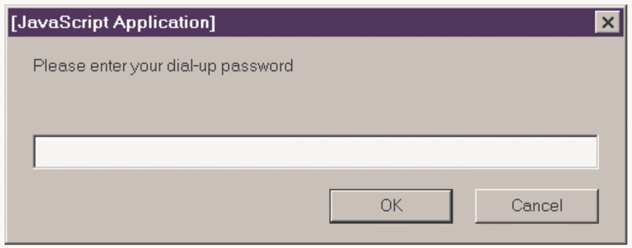
- Buggy browsers: more features ⇨ more bugs
- Netscape’s server push and client pull ⇨ bandwidth
- Helper applications and browser plugins may create security holes (executable commands)

Social Engineering (1/2)

- “There is a problem with your account. Please change your password to NowSafe and await further instructions.”
- “There is a problem with your account and we are unable to bill your credit card. Please enter your credit card number and expiration date and click the SUBMIT button.”
- “We have detected that you are running an out-of-date version of this web browser software. Please click on this URL to download a new version of the software, then run the program called SETUP.EXE to install it.”

Social Engineering (2/2)

```
<SCRIPT>
password = prompt("Please enter your dial-up password", "");
</SCRIPT>
```



Some Browser Flaws

- Predictable random numbers
- Applets can open connections to any host
- Automatically fill user environment info into forms and submit it
- Send hidden email in the name of the user
- Run any program stored on the user's computer
- DNS spoofing attacks on applets

JavaScript Security

- DOS attacks
  - open new browser or alert windows in a loop
  - calculating Fibonacci numbers
  - swap space (create large strings in a loop)
- Running scripts cannot be stopped
- Spoofing (official-looking windows)
- Access to browser history
- Automatic sending of emails
- Monitor other browser windows
- Change status line

Well-trusted Hosts ?

- WWW.MICROS0FT.COM (O <=> 0)
- MICROSOFT.CO.FI
- www.microsoft.co.../setup.exe (i.e., www.microsoft.com.attacker.org/hacker/setup.exe)
- ActiveX
  - Authenticode is no solution
  - run any program without a sandbox
  - malicious code ? buggy code ? viruses ?

Privacy

- User-tracking via cookies
- How much is stored in the log files ?
- Does the web server have a published privacy policy ?
- eTrust: develop standards for online privacy

⇨ Anonymizers

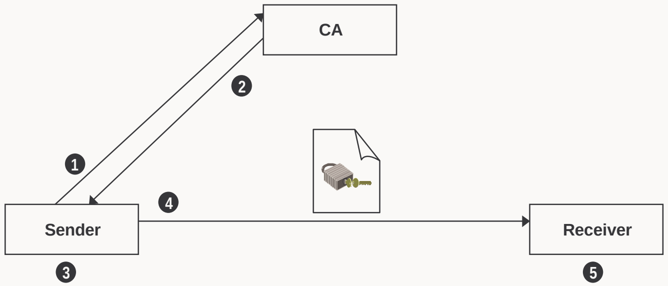
Certificates

- Typically based on public key cryptography (RSA, PGP, ...)
- User generates a key pair
  - private key is kept private (diskette, chipcard)
  - public key is registered with a certification authority (CA) => certificate
- Certificates are NOT people => certification requires identification
- Many different certificate types (most popular X.509v3)

Public Key Infrastructures

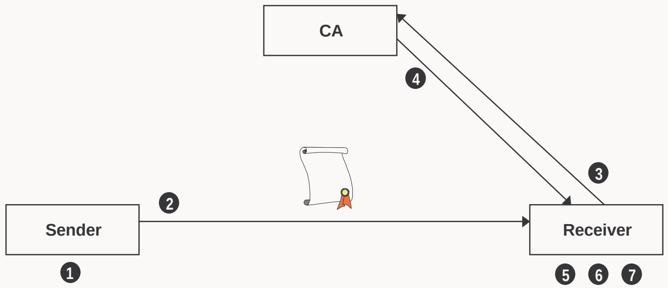
- CA must be trusted
- CA must maintain and clients must check certificate revocation lists (CRLs)
- Communication with the CA must be secure
- CA must digitally sign replies
- How do I get a CA's certificate for my browser ?
- Cross-certification (chains of trust)

Public Key Encryption



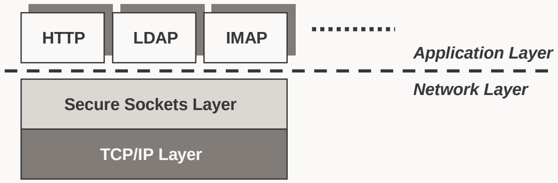
(1) request receivers's certificate
(2) send certificate
(3) check certificate's validity
(4) get receiver's public key from certificate, encrypt message with receiver's public key and send E(M)
(5) decrypt message with private key D(E(M)) = M

Digital Signature



(1) create message digest (e.g., MD5) and encrypt with private key D(h(M))
(2) send M + D(h_1(M)) - S=D(h_1(M)) is the digital signature of the message
(3) request sender's certificate
(4) send certificate
(5) check certificate's validity
(6) compute h_2(M) and decrypt S with sender's public key E(S)=E(D(h_1(M)))
(7) message is authentic and unchanged if h_2(M) is equal to E(S)

Secure Sockets Layer (SSL)



- SSL defined by Netscape => Transport Layer Security (TLS) Protocol
- Provides authentication and non-repudiation of servers and clients, data confidentiality and data integrity
- Uses separate keys and algorithms for encryption, authentication and data integrity (separation of duties)
- Certificate-based authentication
- Protection against replay and man-in-the-middle
- Protocol agnostic

SSL behind the Scenes

Client	Server
• Send ClientHello message (SSL version, ciphers, random data, session id)	
	• Send ServerHello (SSL version, cipher chosen, random data, session id) • Send certificate
• Check server's certificate,generate pre-master secret based on the random data) and send it to server (encrypted with server's key)	
	• Request client certificate (optional) => additional handshake
• Generate master secret out of premaster secret (same series of steps like the server) • Master secret => generate session keys (symmetric encryption)	• Generate master secret out of premaster secret (same series of steps like the client) • Master secret => generate session keys (symmetric encryption)

Pretty Good Privacy (PGP)

- Protection for email and files
- Hybrid encryption system
  - RSA (public key) for key management
  - IDEA (symmetric) for bulk encryption of data
- Management and certification of public keys
  - keys never expire
  - compromised key => keyholder must distribute a revocation certificate
  - key validation trough a web of trust: each user can certify any key (key signing parties)
  - public key distribution by key servers, attached to email, on web page, ...

Electronic Money

- Confidentiality
- Exactly-once semantics (double-spending)
- Scalability (generation, checks, security)
- Anonymity (tracing of purchases)
- Who generates the money ? Who checks its validity ?
- How do I get electronic money ?

Macro- vs. Micropayment

- High security level
- Non-neglectable transaction costs
- T/min low
- Higher amounts

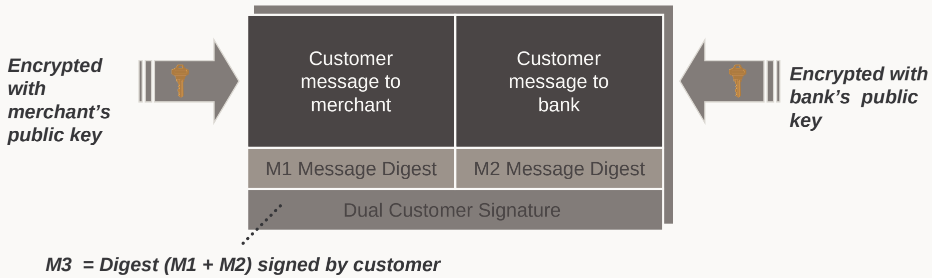
- Limited security level
- Very low transaction costs
- T/min high
- Low amounts (< 1$)

Secure Electronic Transaction (SET)

- Joint standard of MasterCard and Visa
  - confidential transmission
  - authentication of involved parties
  - integrity of payment instructions
- No US export restrictions
  - strong cryptography but cannot be used to encrypt arbitrary texts

SET behind the Scenes

- SET purchase request has 2 parts
  - one for the merchant, one for the bank
  - ⇨ bank does not know about purchased goods, merchant has no credit card info



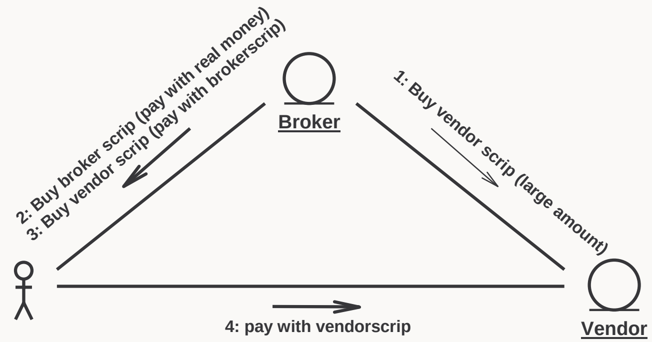
Millicent

- Developed by DEC for small purchases
- Scrip (“digital cash”)
  - models an account a customer has established with a vendor or a broker
  - balance is encoded in the scrip with a proof of correctness (digital signature) for the scrip’s value
  - can be checked locally for correctness (=> no central server !)
  - only valid for a specific vendor or broker
- Scrip security levels
  - in the clear
  - private and secure
  - secure without encryption

Millicent Roles

- Broker
  - buys larger amounts of vendorscrip from many vendors (real-money transaction)
  - sells brokerscrip and vendorscrips to customers (on request)
- Customer
  - buys brokerscrip from a broker (real-money transaction)
  - uses brokerscrip to buy vendorscrip for a specific vendor => payment at the vendor who issued this vendorscrip
  - has a few of accounts with some brokers
- Vendor
  - sells products and accepts its own vendorscrip as payment
  - long-lasting accounts with a few brokers

Payment with Millicent



- Payment
  - customer sends scrip
  - vendor/broker checks validity of scrip
  - if scrip is OK, generates new scrip with reduced balance
  - return reduced scrip as change

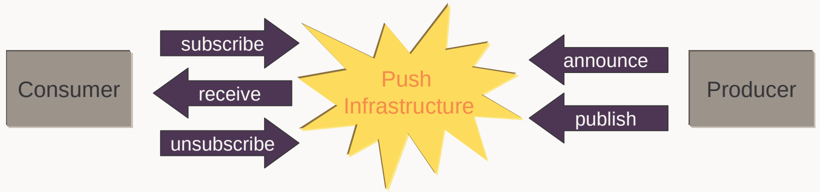
E-Commerce Problems

- Where/how to pay taxes?
- Money transfer (=> national bank)?
- Customer rights
  - Cancellation?
  - Which laws are applied?
- Legal in one country / illegal in another one?

Current Situation

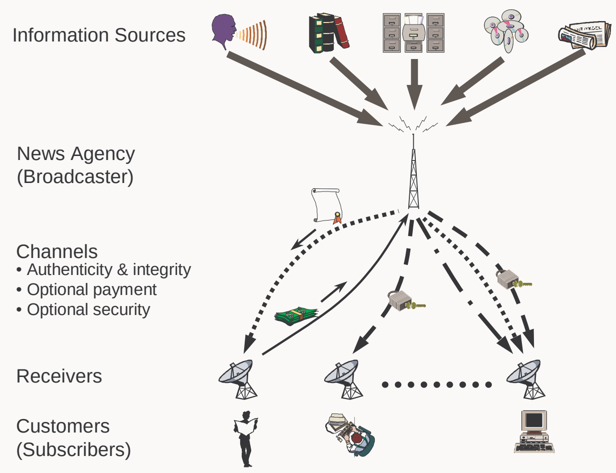
- Unskilled users: Internet = WWW + Email
- Quality of information found/retrieved proportional to knowledge/skills
- On-demand, user-initiated interaction
- Information passively waits for users

A push towards the future



- Users
  - select among information channels
  - subscribe to some channels
- Pull => push
- New interaction pattern: push/pull mix

A Sample Push Scenario



Benefits of Push Systems

- Discovery of information
- Timeliness of information / announcement
- Focused information
- User customized information
- Provider-side information tailoring
- Traffic reduction (collocation with ISPs)
- "Usenet News of the 90ties"

Why not use ....

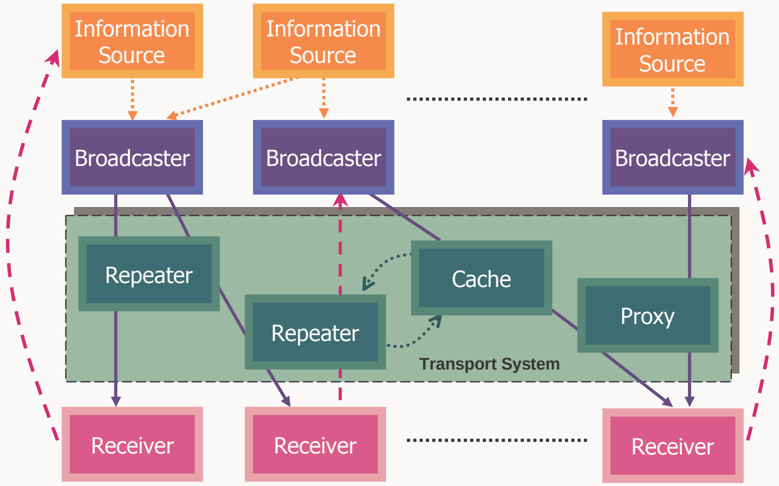
- Email (SMTP)
  - 1:1 ⇨ n recipients: data is duplicated and transmitted n times (disk space, bandwidth)
  - Interaction? Security? E-commerce? Mobile Code?
- Usenet News (NNTP)
  - n:m => lots of data that is not needed (full newsfeed ~400kBit/s = 2.5GB/day!)
  - Security? Authentication? E-commerce? Mobile Code?

... or why not use ...

- WWW is “passive”

- WWW + Email
  - Push (email) / pull (WWW)
  - Scalability problems of email
  - Can exploit existing infrastructure
  - Lack of integration
  - Mobile code? Customization and filtering?

Push System Component Model



Broadcasting Strategies

- Multicast - limited access

- Client pull - robust, simple, scales well, frequency? notification?

- Server push - directory of subscribers, scalability (seq.) ?

- Hybrid approaches - push/pull mix

Comparison (Components)

System	Channel	Broadcaster	Comm. Paradigm	Transport System
Castanet	✓	✓	pull	R, C, P
PointCast	✓	CBF	pull, lim. push	cache
BackWeb	✓	✓	pull & push	cache
Webcasting	✓	---	pull	---
WebCanal	✓	✓	push	---
Intermind	✓	---	pull	---

Comparison (Features)

System	Back-channel	Pushlets	Update Strategy	Filtering	Scalability	Receiver Update	Data Security
Castanet	plugin	✓	diff. (byte)	---	high	✓	high
PointCast	---	limited	?	limited	low–medium	✓	---
BackWeb	✓	✓	diff. (byte)	✓	medium–high	---	high
Webcasting	external	✓	diff. (file)	---	high	✓	low
WebCanal	R = B	browser	diff. (file)	---	low-medium	---	---
Intermind	external	browser	?	limited	medium-high	---	---

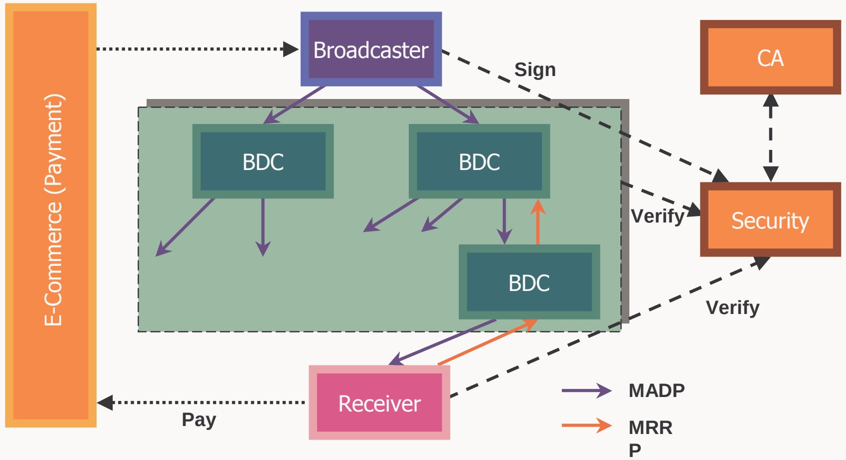
Some Highlights

- Castanet
  - Java-based
  - Distribution and Replication Protocol (DRP)
- Microsoft Webcasting
  - Channel Definition Format (XML DTD)
- WebCanal
  - Multicast

Problems of current Push Systems

- Repeatedly poll broadcaster for updates
- WWW content types, Java, scripting languages
- Not portable: receiver, broadcaster, protocols, etc.
- Scalability problems
- Inadequate security
- No support for e-commerce / payment

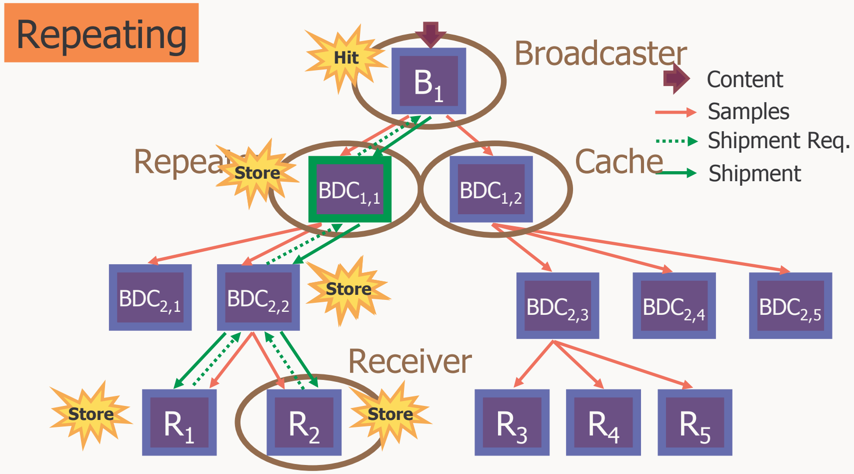
The MINSTREL Push System



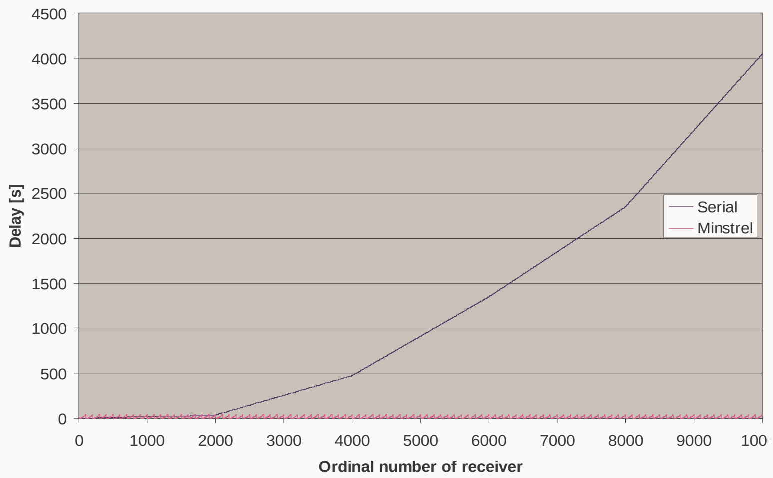
Minstrel's Goals

- Active push distribution to large user groups
- Scalability (users, network bandwidth)
  - Off-line receivers
  - Distribution algorithm (network/server load)
  - Caching Infrastructure
  - Publication of available Channels
- Authenticity and integrity of information
- Payment methods and business models
- Static and executable content (security !)

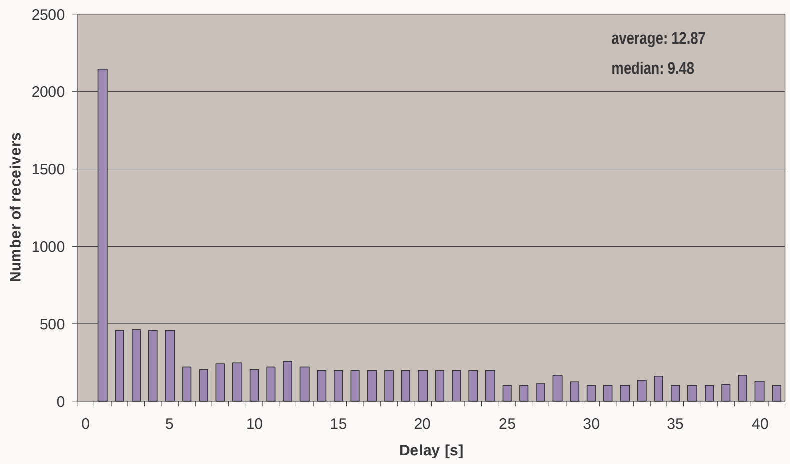
Hybrid Broadcasting



Delay: Minstrel vs. Serial



Distribution of Delays



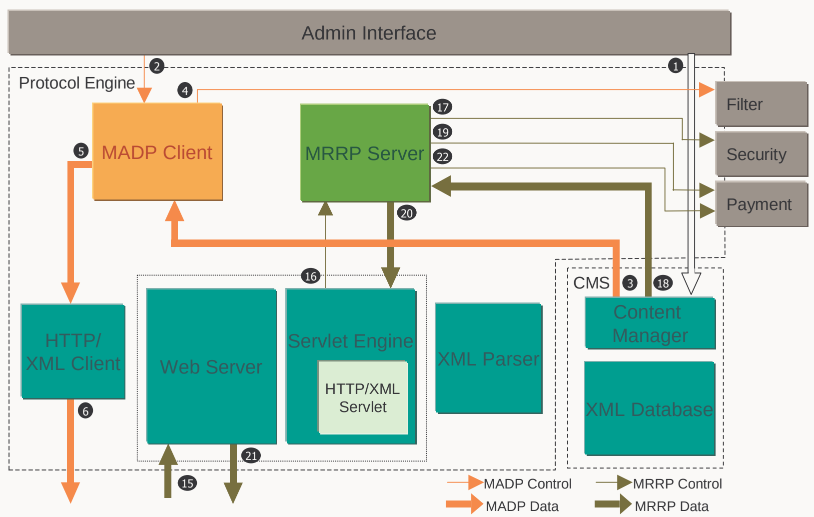
Technology Used

- MADP & MRRP: XML over HTTP
- HTTP for Communication Purposes
  - well known and widely deployed
  - secure (SSL)
  - works across Firewalls
  - COTS are available (HTTP Servers)
- XML for Protocol Messages
  - open
  - extensible
  - COTS are available (XML Parsers, DBs)

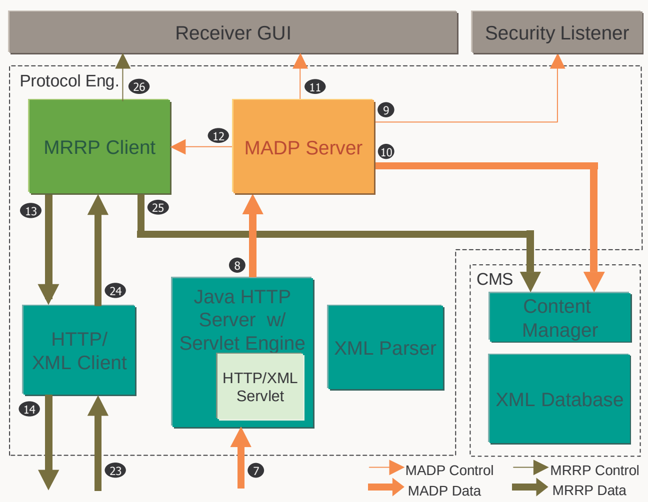
Minstrel Protocol Messages


```
<?xml version="1.0"?>

<Sample sid="k32a5ds64f8sdf9ak4fhf2sf"
            priority=1>
   <Offer oid="itu435gf6b3vkjr234ei6gk2c"
            timestamp="Sat May 13 08:01:21 GMT+02:00 2000"
            vendorId="bk4sfas34vbdf98jkgiut845j"
            channelId="bk4sfas34vbdf98jkgiut845j"
            validFrom="13.05.2000"
            validTo="14.05.2000"
            description="Satellite Image for 13.05.200 available"
            price=0.5
            currency="USD">

      <ProductInfo pid="54blk4kgkj459grofbq35yc"
                  name="sat13052000.jpg"
                  version=1.0 />
   </Offer>

   <Cargo cid="jhv5cx143mb62nmk8itu"
            contentUrl="http://hpp20:8080/Content/Images" />
</Sample>
```


Broadcaster Architecture



Receiver Architecture



Push Systems vs. Event-based Systems

	Push Systems	Event-based Systems
Purpose	timely data distribution	event notification
Participant roles	asymmetric	symmetric
Advertisement policy	simple advertisement (channel)	expressive advertisement language
Subscription policy	simple subscription (channel)	expressive subscription language
Frequency of events	low to medium	high
Number of events	low to medium	high
Payload size	large	small
P/C interconnection	static channels & static producers	dynamic binding to producers
Event grouping	channel	event patterns
Filtering	reduce data transmission req.	reduce number of events

Pushing a dead Horse ?

- 1996: big media hype
- 1999: renaissance
- EU projects (IST program)
  - OPELIX (An Open Personalized Electronic Information Commerce System)
  - MOTION (MObile Teamwork Infrastructure for Organisations Networking)

Ahead

Exam

- 18.12.2002
- Room? ⇨ Check website
- Time? ⇨ Check website
- Closed book

GOOD LUCK !