Zufällige UUIDs? Echter Zufall?

http://webnz.com/robert/true_rng.html

The usual method is to amplify *noise* generated by a *resistor* (Johnson noise) or a semi-conductor *diode* and feed this to a comparator or Schmitt trigger. If you sample the output (not too quickly) you (hope to) get a series of bits which are statistically independent.

www.random.org

Random.org offers *true random numbers* to anyone on the *Internet*.

At this time, there are two ways to obtain the random numbers: via the *World Wide Web* and via the *CORBA* server.

Computer engineers chose to introduce randomness into computers in the form of *pseudo-random number generators*. As the name suggests, pseudo-random numbers are not truly random. Rather, they are computed from a mathematical formula or simply taken from a precalculated list.

Random numbers are used for computer games but they are also used on a more serious scale for the generation of cryptographic keys and for some classes of scientific experiments. For *scientific experiments*, it is convenient that a series of random numbers can be replayed for use in several experiments, and pseudo-random numbers are well suited for this purpose. For *cryptographic use*, however, it is important that the numbers used to generate keys are not just seemingly random; they must be truly *unpredictable*.

The way the random.org random number generator works is quite simple. A radio is tuned into a frequency where nobody is broadcasting. The *atmospheric noise* picked up by the receiver is fed into a Sun SPARC workstation through the microphone port where it is sampled by a program as an eight bit mono signal at a frequency of 8KHz. The upper seven bits of each sample are discarded immediately and the remaining bits are gathered and turned into a stream of bits with a high content of entropy. *Skew correction* is performed on the bit stream, in order to insure that there is an approximately even distribution of 0s and 1s.

The skew correction algorithm used is based on transition mapping. Bits are read two at a time, and if there is a *transition between values* (the bits are 01 or 10) one of them - say the first - is passed on as random. If there is no transition (the bits are 00 or 11), the bits are discarded and the next two are read. This simple algorithm was originally due to *Von Neumann* and completely eliminates any bias towards 0 or 1 in the data. [***Why??***]

Web Interface to True Random Numbers

http://www.random.org/nform.html

Smallest value 1, largest value 100, format in 5 columns:

37	36	10	44	94
79	12	61	43	100
63	37	27	30	30
41	96	57	19	83

Flip a Coin

http://www.random.org/flip.html

A Kr85-based Random Generator

http://www.fourmilab.ch/hotbits/ ...by John Walker

The *Krypton-85* nucleus (the 85 means there are a total of 85 protons and neutrons in the atom) spontaneously turns into a nucleus of the element *Rubidium* which still has a sum of 85 protons and neutrons, and a *beta particle (electron) flies out*, resulting in no net difference in charge. What's interesting, and ultimately useful in our quest for random numbers, is that even though we're absolutely certain that if we start out with, say, 100 million atoms of Krypton-85, 10.73 years later we'll have about 50 million, 10.73 years after that 25 million, and so on, there is *no way even in principle* to predict when a given atom of Krypton-85 will decay into Rubidium.

So, given a Krypton-85 nucleus, there is *no way whatsoever to predict when it will decay.* If we have a large number of them, we can be confident half will decay in 10.73 years; but if we have a single atom, pinned in a laser ion trap, all we can say is that is there's even odds it will decay sometime in the next 10.73 years, but as to precisely when we're fundamentally quantum clueless. The only way to know when a given Krypton-85 nucleus decays is after the fact--by detecting the ejecta.

This *inherent randomness* in decay time has profound implications, which we will now exploit to generate random numbers. For if there's no way to know when a given Krypton-85 nucleus will decay then, given an collection of them, there's no way to know when the next one of them will shoot its electron bolt.

Since the time of any given decay is random, then *the interval between two consecutive decays is also random*. What we do, then, is measure a pair of these intervals, and *emit a zero or one bit based on the relative length of the two intervals.* If we measure the same interval for the two decays, we discard the measurement and try again, to avoid the risk of inducing bias due to the resolution of our clock.

To create each random bit, we wait until the first count occurs, then measure the time, T1, until the next. We then wait for a third pulse and measure T2, yielding a pair of durations... if *T1 is less than T2* we emit a *zero* bit; if T1 is *greater* than T2, a *one* bit. In practice, to avoid any residual bias resulting from non-random systematic errors in the apparatus or measuring process consistently favouring one state, the sense of the comparison between T1 and T2 is reversed for consecutive bits.

For example, you might worry about the fact that the intensity of the radiation source is *slowly decreasing over time*. Krypton-85's *10.73 year half-life* isn't all that long. One half-life in the future, we'll measure T1 and T2 intervals, on the average, *twice as long as today.* This means, then, that even on consecutive measurements there is a small bias in favour of T2 being longer than T1. How serious is this?

...this means T2 will be, on average, 10^{-14} seconds longer than T1. The crystal oscillator which provides the time base for the computer making the measurement is only *accurate* to 100 parts per million, or *one part in ten thousand*, and thus can induce errors ten million times as large as those due to the slow decay of the source. (This is, again, unlikely to be a real problem because most computer clocks, while prone to drifting as temperature and supply voltage vary, do not change significantly on the millisecond scale. Still, *jitter due to where the clock generator happens to trigger on the oscillator waveform* will still dwarf the effects of decay of the source during one measurement.)

You request HotBits by filling out and transmitting a *request form*, which is sent by your *World-Wide Web browser* in HTTP to our Web server, www.fourmilab.ch. Your request form is processed by a CGI program written in Perl which, after validating the request, forwards it in HTTP format to a dedicated HotBits server machine which is connected to the HotBits generation hardware via the COM1 port.

To provide better response, the dedicated HotBits server machine *maintains an inventory* of two million (256 kilobytes) random bits, and services requests from this inventory whenever possible.

To detect radioactive decay events, I use a commercial radiation monitor which contains a *Geiger-Müller tube detector.*

Radiation is ubiquitous in our fair universe... Although *background radiation* can be used, you either have to not need very many random bits or else be very patient, since background radiation counts only occur every few seconds. To crank up the bit generation rate to something usable for a server accessible on the Internet, we need a *radiation source* more intense than background radiation.

... a 60 microcurie Jordan Nuclear *Krypton-85 source capsule*, model BB-0005. The capsule is about two centimetres long, and has a foil window on the left side through which the radiation emerges. A better choice would be a 10 microcurie *Cesium-137* check source. In most locations, no license is required to obtain such a check source, which can be *ordered through the mail* from dealers in such gadgets.

Another alternative is to visit a *shop catering to rock collectors* and buy a specimen of a *Uranium*-bearing ore such as Carnotite or Pitchblende.

The generator doesn't have to be anywhere near the computer. In fact, it's located three floors down in a converted 70,000 litre subterranean water cistern with metre-thick concrete walls... No need to worry about stray radiation zipping around the computer room!

The randomX package for Java
http://www.fourmilab.ch/hotbits/source/randomX/randomX.html

Class randomX.randomHotBits

// Implementation of a randomX-compliant class which obtains genuine
// random data from John Walker's HotBits radioactive decay random
// sequence generator.

public byte nextByte()

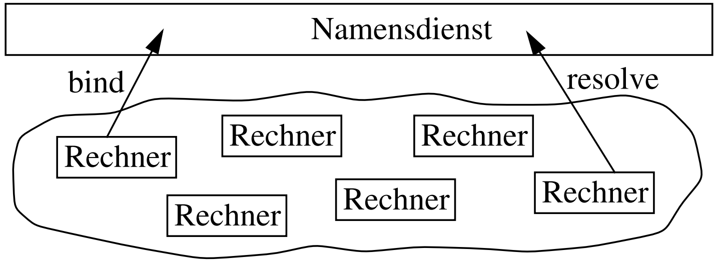
// Get next byte from generator.

http://www.fourmilab.ch/hotbits/generate.html

The following hexadecimal data are the *random bytes you requested*:

```
AFBD9001692FE32805C8AAB7D49BA069D1F641987ED8C28865B2FCF23A9B91DF
927F52E4083CCC73C1E3200C806F5B31D954117ADCAD7B6A0CA7814B1540CE60
065F62E8FB0018F04C86159085E46465987633135232D060E4A255BABAE26D68
```


Verteilte Namensverwaltung



logisch ein einziger Dienst; tatsächlich verteilt realisiert

- Jeder Kontext wird (logisch) von einem autonomen *Nameserver* verwaltet
  - ggf. ist ein Nameserver für mehrere Kontexte zuständig
  - ggf. Aufteilung / Replikation des Nameservers
    --> höhere Effizienz, Ausfallsicherheit

- Typischerweise *hierarchische Namensräume*
  - entsprechend strukturierte Namen
  - entsprechend kanonische Aufteilung der Verwaltungsaufgaben
  - Zusammenfassung Namen gleichen Präfixes vereinfacht Verwaltung

- Typisch: *kooperierende* Nameserver, die den gesamten Verwaltungsdienst realisieren
  - hierzu geeignete Architektur der Server vorsehen
  - Protokoll zwischen den Nameservern
    (für Fehlertoleranz, update der Replikate etc.)
  - Dienstschnittstelle wird i.a. durch lokale Nameserver realisiert

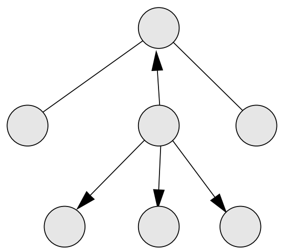
bzw. “user agent”

- *Annahmen*, die Realisierungen i.a. zugrundeliegen:
  - *lesende* Anfragen viel häufiger als schreibende (“Änderungen”)
  - *lokale* Anfragen (bzgl. eigenem Kontext) dominieren
  - seltene, temporäre *Inkonsistenzen* können toleriert werden

ermöglicht effizientere Realisierungen
(z.B. Caching, einfache Protokolle...)

Namensinterpretation in verteilten Systemen

- Ein Nameserver kennt den Nameserver der nächst höheren Stufe



- Ein Nameserver kennt alle Nameserver der untergeordneten Kontexte (sowie deren Namensbereiche)

- Hierarchiestufen sind i.a. klein (typw. 3 oder 4)

- Blätter verwalten die eigentlichen Objektadressen und bilden die Schnittstelle für die Clients

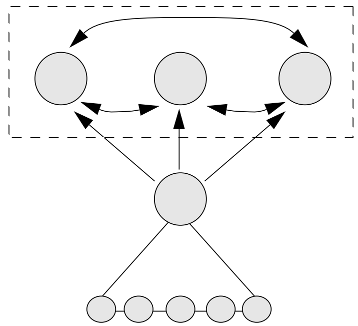
- Nicht interpretierbare Namen werden an die nächst höhere Stufe weitergeleitet (bei strukturierten Namen!)

Broadcast

- falls zuständiger Nameserver unbekannt
  ("wer ist für XYZ zuständig?" oder: "wer ist hier der Nameserver?")
- ist aufwendig, falls nicht durch Hardware etc. unterstützt (wie z.B. bei LAN)
- nur in begrenzten Kontexten anwendbar

Replikation von Nameservern

- Zweck: Erhöhung von Effizienz und Fehlertoleranz
- Vor allem auf höherer Ebene relevant
  - dort viele Anfragen
  - Ausfall würde grösseren Teilbereich betreffen



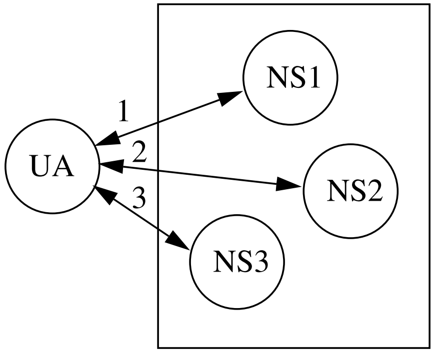
- Server kennt alle übergeordneten Server
- Broadcast an ganze Servergruppe, oder Einzelnachricht an "nächsten" Server; anderen Server erst nach Ablauf eines Timeouts befragen

- Replizierte Server konsistent halten
  - ggf. nur von Zeit zu Zeit gegenseitig updaten (falls veraltete Information tolerierbar)
  - Update auch dann sicherstellen, wenn einige Server zeitweise nicht erreichbar sind (periodisches Wiederholen von update-Nachrichten)
  - Einträge mit Zeitstempel versehen --> jeweils neuester Eintrag dominiert (global synchronisierte Zeitbasis notwendig!)

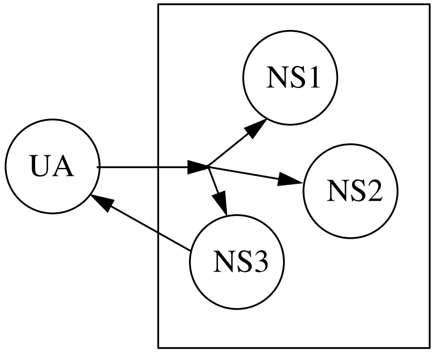
- Symmetrische Server / Primärserver-Konzept:
  - *symmetrische Server*: jeder Server kann updates initiieren
  - *Primärserver*: nur dieser nimmt updates entgegen
    - Primärserver aktualisiert gelegentlich "read only" Sekundärserver
    - Rolle des Primärservers muss im Fehlerfall von einem anderen Server der Gruppe übernommen werden

Strukturen zur Namensauflösung

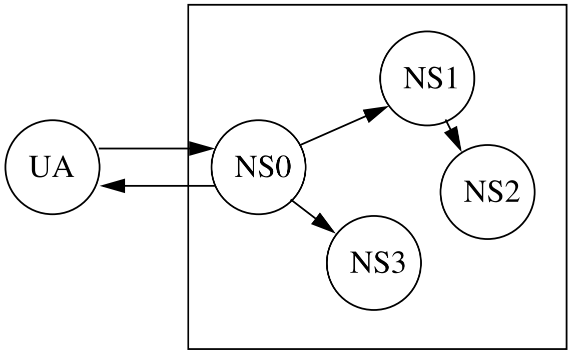
- User Agent (UA) bzw. “Name Agent” auf Client-Seite
  - hinzugebundene Schnittstelle aus Bibliothek, oder
  - eigener Service-Prozess



Iterative Navigation: NS1 liefert Adresse eines anderen Nameservers zurück bzw. UA probiert einige (vermutlich) zuständige Nameserver nacheinander aus



Multicast-Navigation: Es antwortet derjenige, der den Namen auflösen kann (u.U. auch mehrere)



“Rekursive” Namensauflösung, wenn ein Nameserver ggf. den Dienst einer anderen Ebene in Anspruch nimmt

Serverkontrollierte Navigation: Der Namensdienst selbst in Form des Serververbundes kümmert sich um die Suche nach Zuständigkeit

Caching von Bindungsinformation

- Zweck: Leistungsverbesserung, insbesondere bei häufigen nichtlokalen Anfragen

(a) Abbildung Name --> Adresse des *Objektes*

(b) Abbildung Name --> Adresse des *Nameservers* der tiefsten Hierarchiestufe, der für das Objekt zuständig ist

- Zuordnungstabelle wird lokal gehalten
- vor Aufruf eines Nameservers überprüfen, ob Information im Cache
- Information könnte allerdings veraltet sein!
- Platz der Tabelle ist beschränkt --> unwichtige / alte Einträge verdrängen
- Neue Information wird als Seiteneffekt einer Anfrage eingetragen

---

- Vorteil von (b): Inkonsistenz aufgrund veralteter Information kann vom Nameservice entdeckt werden
  - veralteter Cache-Eintrag kann transparent für den Client durch eine automatisch abgesetzte volle Anfrage ersetzt werden
- Bei (a) muss der *Client* selbst falsche Adressen *beim Zugriff* auf das Objekt erkennen und behandeln

---

- Caching kann bei den Clients stattfinden (z.B. im RPC-Laufzeitsystem) und / oder bei den Nameservern

Beispiele für Namensdienste

- Domain Name System (DNS) im Internet
  - in der UNIX-Welt oft eingesetzte Implementierung: BIND ("Berkeley Internet Domain Name")

- Neben DNS als Quasistandard existiert u.a. die internationale X500-Norm ("CCITT / ISO-OSI directory service") als globaler Namensdienst.
  - Normung der Struktur der Einträge und der Protokolle
  - OSF-DCE nutzt X500 als zellübergreifenden Namensservice
  - neue Attribute definierbar (Name plus Syntaxdefinition in ANS.1)

- LDAP (Lightweight Directory Access Protocol)

- Network Information Service (NIS)
  - entwickelt von Sun Microsystem (ursprünglich: "Yellow Pages")
  - hauptsächlich zur Verwaltung von Dateizugriffsrechten in lokal vernetzten Systemen
  - später erweitert zur Verwaltung von Benutzern, Passwörtern, Diensten...
  - basiert auf RPC
  - nutzt Primär- / Sekundärserverprinzip ("Master" / "Replica Server")

- Portmapper für TCP- oder UDP-basierte Dienste
  - eher rudimentär; nicht verteilt

- Lookup-Service ("LUS") bei Jini und ähnlichen Systemen

Internet Domain Name System (DNS)

- Jeder Rechner im Internet hat eine IP-Adresse
  - 32 Bit lang (bei IPv4), typischerweise als 4 Dezimalzahlen geschrieben
  - Bsp.: 192.130.10.121 (= 11000000.10000010.00001010.01111001)

- Symbolische Namen sind besser
  - z.B. Domain-Namen wie www.nanocomp.uni-cooltown.eu
  - gut zu merken; relativ unabhängig von spezifischer Maschine
  - muss bei *vor* Verwendung bei Internet-Diensten (telnet, ftp, Email, WWW...) in eine IP-Adresse umgesetzt werden
  - Umsetzung in IP-Adresse geschieht im Internet mit DNS

- Domains
  - hierarchischer Namensraum der symbolischen Namen im Internet
  - "Toplevel domains" de, fr, ch, edu, com,... (neu: aero, biz, coop, info, museum, name, pro ; u.U. bald auch eu, arts, firm, rec, shop, web,...)
  - Domains (ggf. rekursiv) gegliedert in Subdomains, z.B.

```
                    eu
            uni-cooltown.eu
  informatik.uni-cooltown.eu
nano.informatik.uni-cooltown.eu
pc6.nano.informatik.uni-cooltown.eu
```

- Für einzelne (Sub)domains bzw. einer Zusammenfassung einiger (Sub)domains (sogenannte "Zonen") ist jeweils ein Domain-Nameserver zuständig
  - primärer Nameserver
  - optional zusätzlich einige weitere sekundäre Nameserver
  - oft sind Primärserver verschiedener Zonen gleichzeitig wechselseitig Sekundärserver für die anderen
  - Nameserver haben also nur eine Teilsicht!

DE-NIC und SWITCH

- Deutsches Network Information Center (www.nic.de)

  - Betrieb des Primary-Nameservers ns.nic.de für die Toplevel-Domain DE
  - Registrierung von Domains unterhalb von DE
  - 1991: Erster freiwilliger Nameserver-Dienst für .de (kurz: DENIC) an der Universität Dortmund
  - August 1993: Gründung des "Interessenverbund Deutsches Network Information Center" (IV-DENIC)
  - Januar 1994: Aufnahme des Betriebs für zunächst drei Jahre an der Universität Karlsruhe
  - Dezember 1996: Gründung der DENIC eG (“eingetragene Genossenschaft”)
  - Juli 1997: Aufnahme des Betriebs am neuen Sitz in Frankfurt

  Die Mitglieder der DENIC eG sind Internet Service Provider (ISP), die ihren Kunden lokale Zugänge zum Internet zur Verfügung

- Anzahl der Rechner

  30.06.1996: Deutschland: 576 296; Europa: 2 910 043
  05.11.1998: Deutschland: 1 404 695; Europa: 7 647 544
  20.05.2000: Deutschland: 1 673 091; Europa:10 910 395

- Anzahl der Domains unterhalb DE

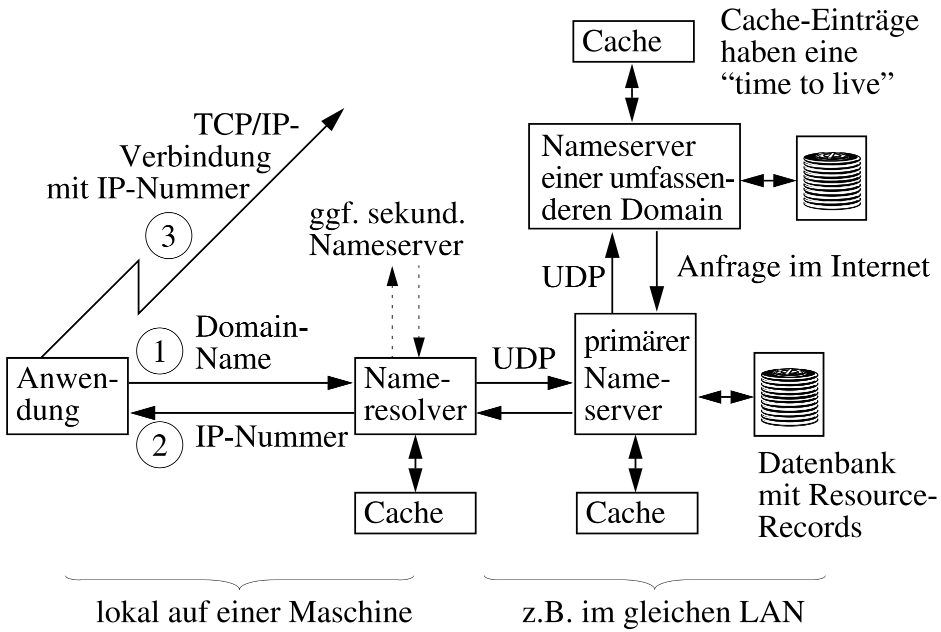
Juli 1996:	20 257	Mai 2000: 2 201 864
Okt. 1998:	223 187	Mai 2001: 4 470 000

- In der Schweiz: SWITCH (www.switch.ch)
  - Gegründet 1987 von der Eidgenossenschaft und 8 Universitätskantonen
  - Verantwortlich für Country Code Top Level Domains “ch” und “li”
  - Primary-Nameserver für diese Domains (Ende 2000: 350 000 Domains)
  - April 2001: 250 600 hosts

Namensauflösung im Internet

- Historisch: Jeder Rechner hatte eine Datei hosts.txt, die jede Nacht von zentraler Stelle aus verteilt wurde

- Jetzt: lokaler Nameresolver, der eine Zuordnungsdatei /etc/hosts für die wichtigsten Rechner enthält, und sich ansonsten an einen seiner nächsten Nameserver wendet
  - IP-Nummern der lokalen Nameserver stehen in der Datei resolv.conf



Resource Records

- Datenbank eines DNS-Nameservers besteht aus einer Menge von Resource-Records, z.B.:

fb22.tu-da.de	IN SOA ...
sys1.fb22.tu-da.de	IN A 130.83.200.63
sys1.fb22.tu-da.de	IN A 130.83.253.12
fb22.tu-da.de	IN MX mailgate.fb22.tu-da.de
www.fb22.tu-da.de	IN CNAME robin.fb22.tu-da.de
ftp.fb22.tu-da.de	IN CNAME robin.fb22.tu-da.de
fb23.tu-da.de	IN NS 130.83.193.77
boss	IN A 130.83.200.17
helga	IN A 130.83.200.39
laserjet	IN A 130.83.201.75

symbolischer Name

IP-Adresse (bei A-records)

- Verschiedene Record-Formate, z.B.:
  - A für "Address"
  - SOA ("Start of Authority"): Parameter zur Zone (z.B. für Caching etc.)
  - MX: forwarding von E-mail ("Mail eXchanger")
  - CNAME ("Canonical Name"): für Spezifikation eines Alias
  - NS: Nameserver für eine Subdomain

- Einige weitere Angaben stehen in anderen Dateien, z.B.:
  - IP-Adresse der übergeordneten Nameserver ("root server")
  - ob Primär- oder Sekundärserver etc.

nslookup

NAME in.named, named

in.named is the Internet domain name server. It is used by hosts on the Internet to provide access to the Internet distributed naming database. See RFC 1034 and RFC 1035 for more details. With no arguments, in.named reads /etc/named.boot for any initial data, and listens for queries on a privileged port.

NAME nslookup - query name servers interactively

nslookup is an interactive program to query Internet domain name servers. The user can contact servers to request information about a specific host, or print a list of hosts in the domain.

```
> sun20
Name:    sun20.nanocomp.inf.ethz.ch
Address:  129.132.33.79
Aliases:  ftp.nanocomp.inf.ethz.ch

> altavista.com
Name:    altavista.com
Addresses:  204.123.2.75, 204.123.2.66,
204.123.2.69

> altavista.com
Name:    altavista.com
Addresses: 204.123.2.66, 204.123.2.69,
204.123.2.75

> cs.uni-sb.de
Name:    cs.uni-sb.de
Addresses:  134.96.254.254, 134.96.252.31
```

Dies deutet auf einen "round robin"-Eintrag hin: Der Nameserver von altavista.com ändert alle paar Minuten die Reihenfolge der Einträge, die bei anderen Nameservern auch nur einige Minuten lang gespeichert bleiben dürfen. Da Anwendungen i.a. den ersten Eintrag nehmen, wird so eine Lastverteilung auf mehrere Altavista-Server vorgenommen!

Router an zwei Netzen

Namen für Orte?

- Ortsbezogene Dienste bei mobile computing interessant

- Positionsbestimmung relativ einfach / billig möglich
    - GPS
    - Mobilfunkzelle

- Aber was nützt i.a. eine Angabe *047° 22' N, 008° 33' E* ?

Posotion ("Adresse")

- Besser wäre vielleicht
    - im Hörsaal X56
    - im Gebäude HIT
    - in der ETH
    - in Zürich
    - in der Schweiz
    - ...

das sind alles Namen!

- was ist deren genaue Bedeutung? ("logischer Ort")
- Approximation durch Polygon aus Ortskoordinaten?
- Struktur?
    - "Topologie"?
    - Hierarchie (Enthaltensein)?
    - Nachbarschaft?

- aber was davon, wenn alles richtig ist?

- Und was ist mit solchen Ortsbezeichnungen?
    - in einem Supermarkt
    - zu Hause

- Und wieso nicht gleich Namen im Raum-Zeit-Kontinuum betrachten?
    - z.B. "Samstag abend im Kino"