Lecture 16

Internet Protokolle II

Practical P2P Systems

Thomas Fuhrmann

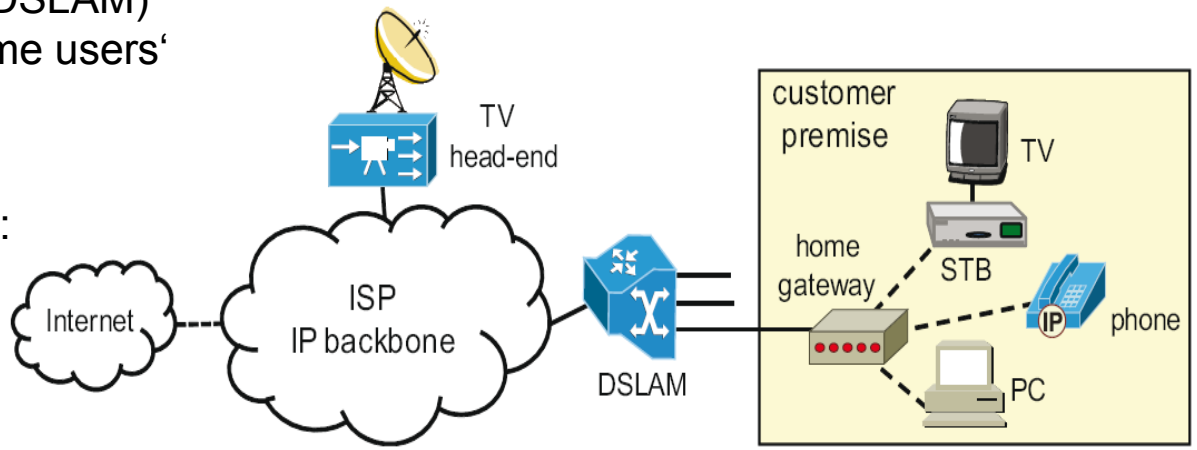
Network Architectures
Computer Science Department
Technical University Munich

Practical P2P Systems

- Filesharing has been and probably still is the predominant use for peer-to-peer technology
  - Use resources at the network‘s edge
  - More robust and much less expensive than centralized servers
  - Much less control about the application (→sharing of copyrighted content)
- Other applications are possible but haven‘t succeeded in practice – except for Skype and video streaming.
- P2P Video Streaming is especially popular in Asia, because the demand for Internet access pushed the market before other multimedia networks were established.

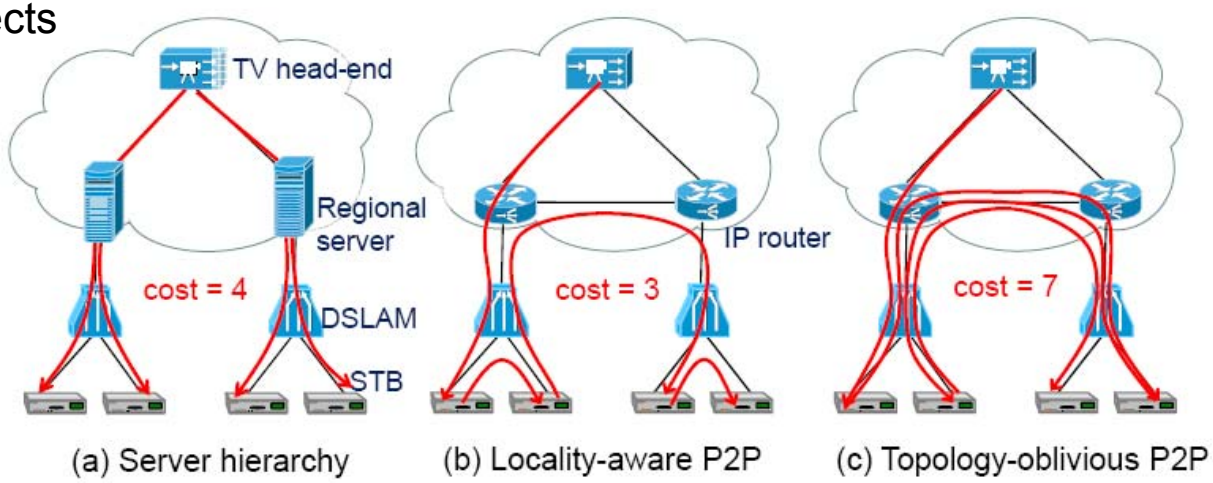
IPTV Architectures

- DSL access multiplexers (DSLAM) aggregate hundreds of home users' traffic.



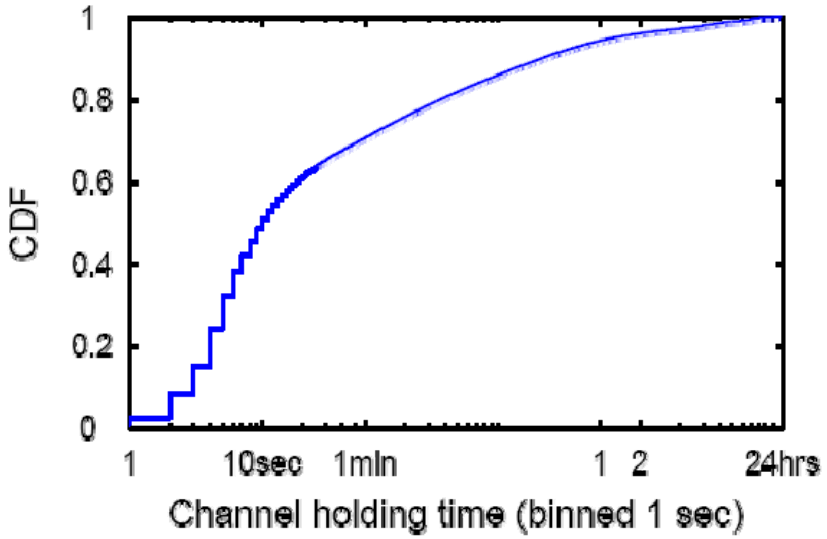
Three alternative architectures:

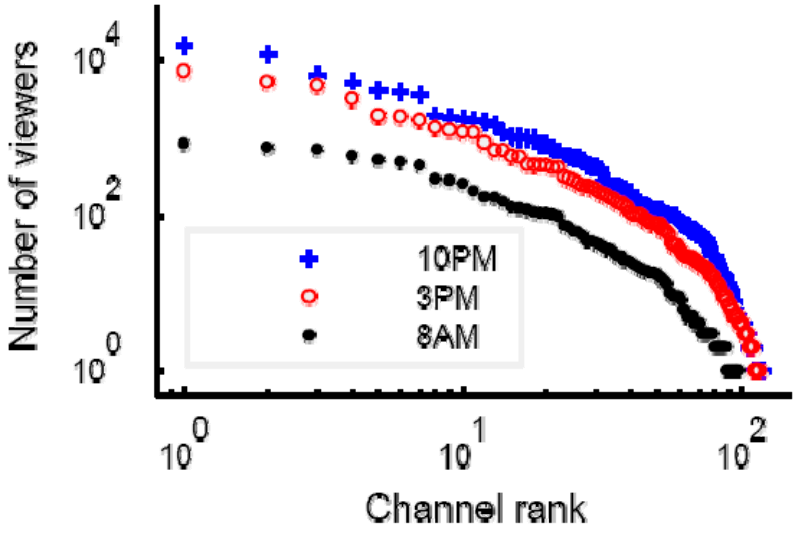
- Video Servers near the DSLAM cache the content from the head-end and serve the customers with unicast streams.
- Native-IP-Multicast connects set-top boxes (STB) with the head-end.
- P2P technology streams the content from the STBs to other STBs.

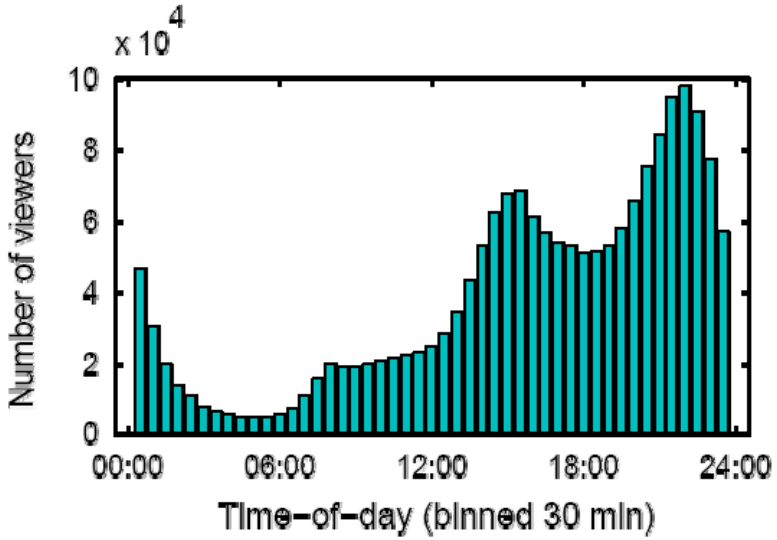


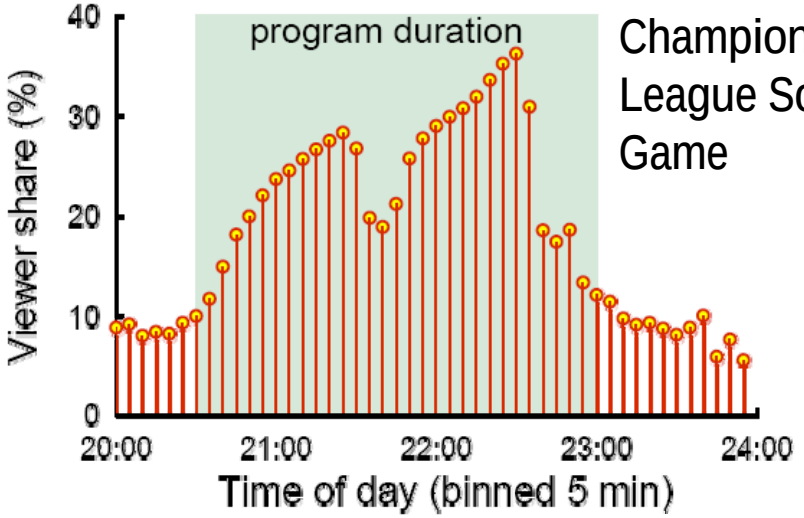
Source: Cha et al. On Next-Generation Telco-Managed P2P TV Architecture, IPTPS 2008

Customers‘ Television Usage Behavior



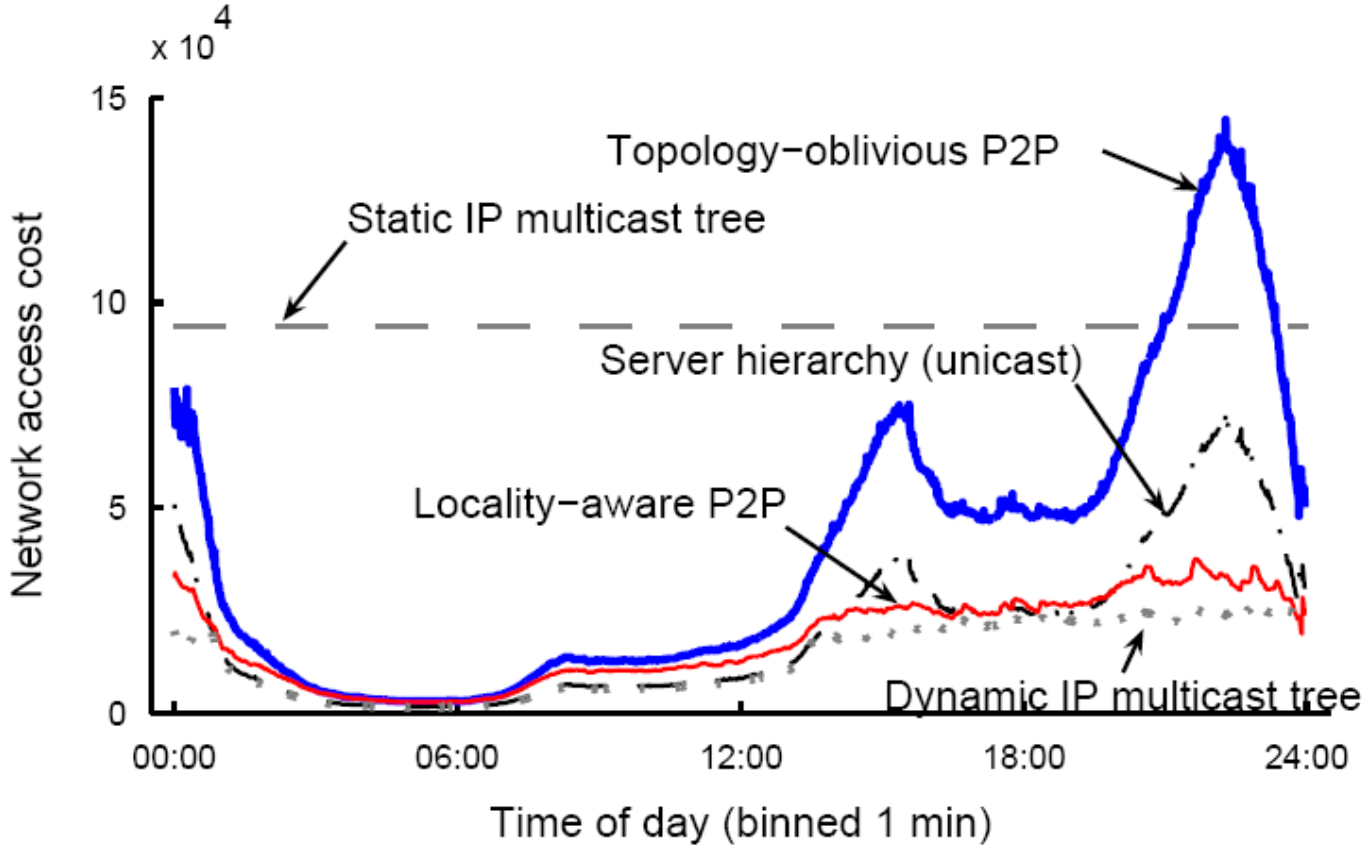






Champions League Soccer Game

Bandwidth Consumption Comparison

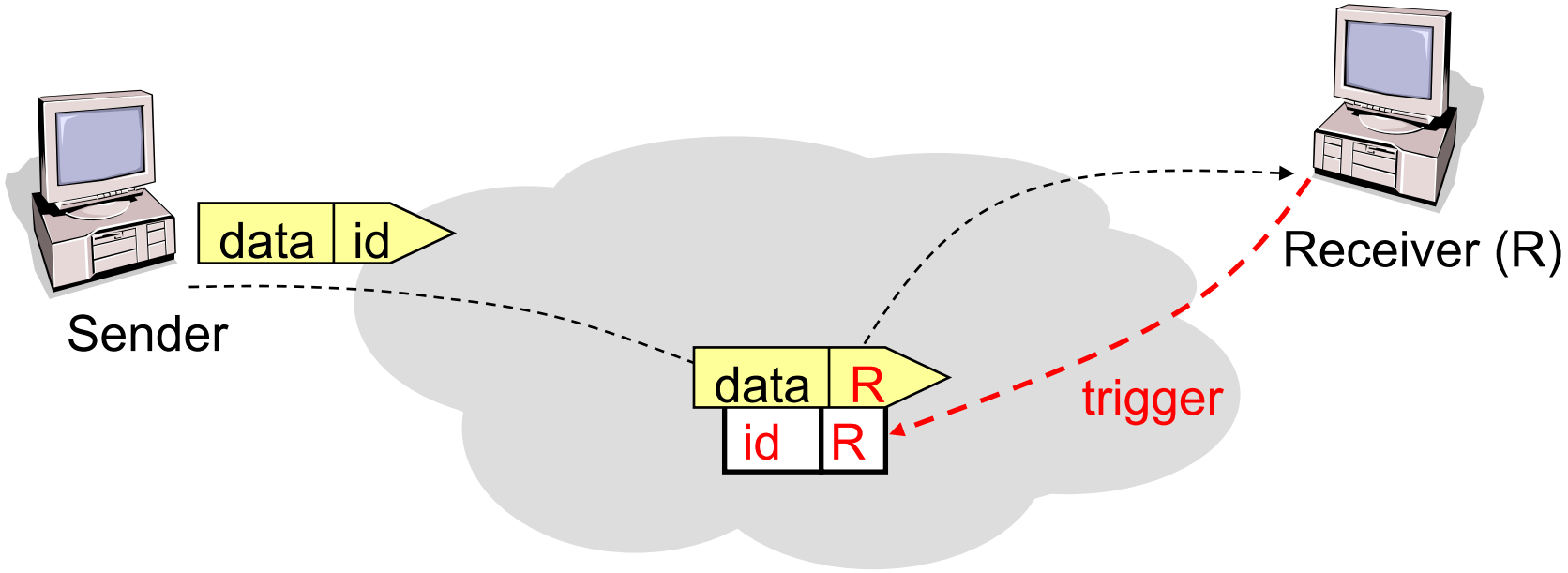


Source: Cha et al. On Next-Generation Telco-Managed P2P TV Architecture, IPTPS 2008

Internet Indirection Infrastructure

i3 – Internet Indirection Infrastructure

- Kommunikationsnetze bewegen Daten von A nach B.
- Verteilte Anwendungen nutzen das Netz, typischerweise im Client-Server Paradigma.
- Was wäre wenn das Netz mehr kann, als nur Daten zu transportieren?

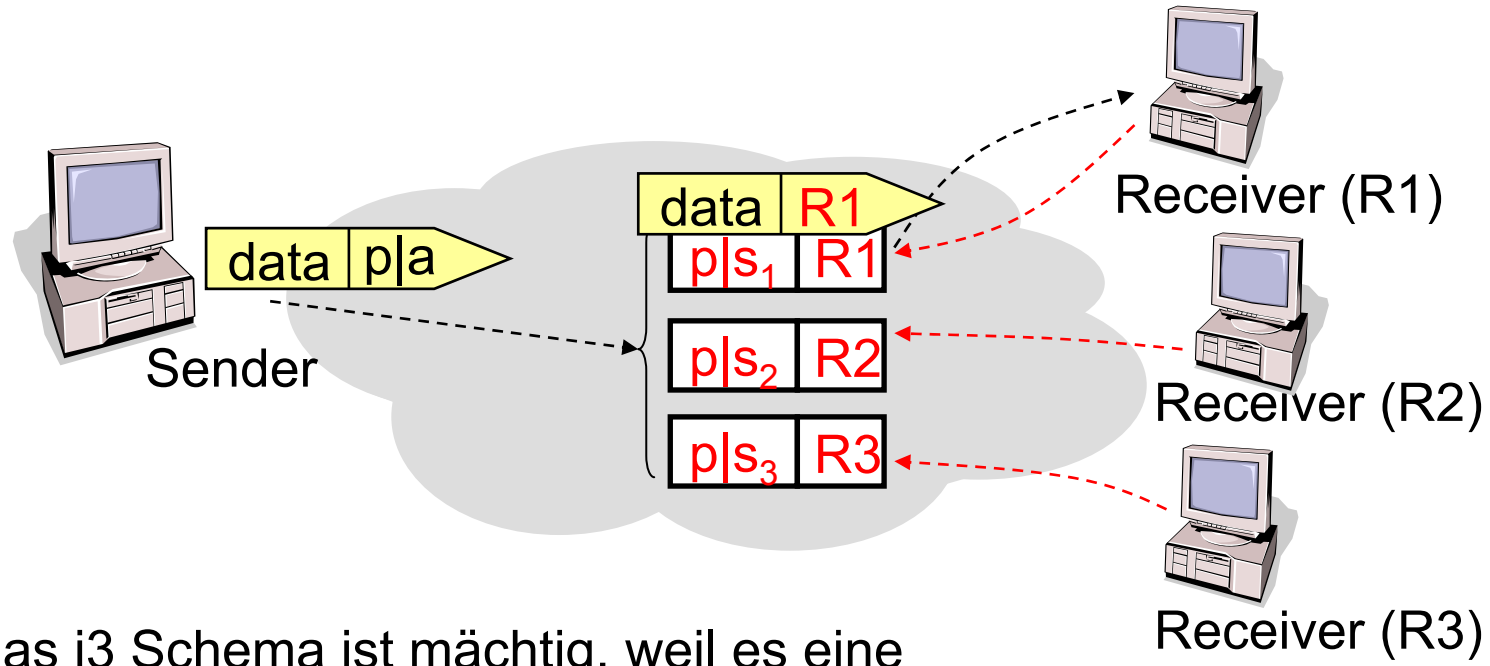


Ion Stoica, et al. Internet Indirection Infrastructure, SIGCOMM 2002

i3 – Internet Indirection Infrastructure (2)

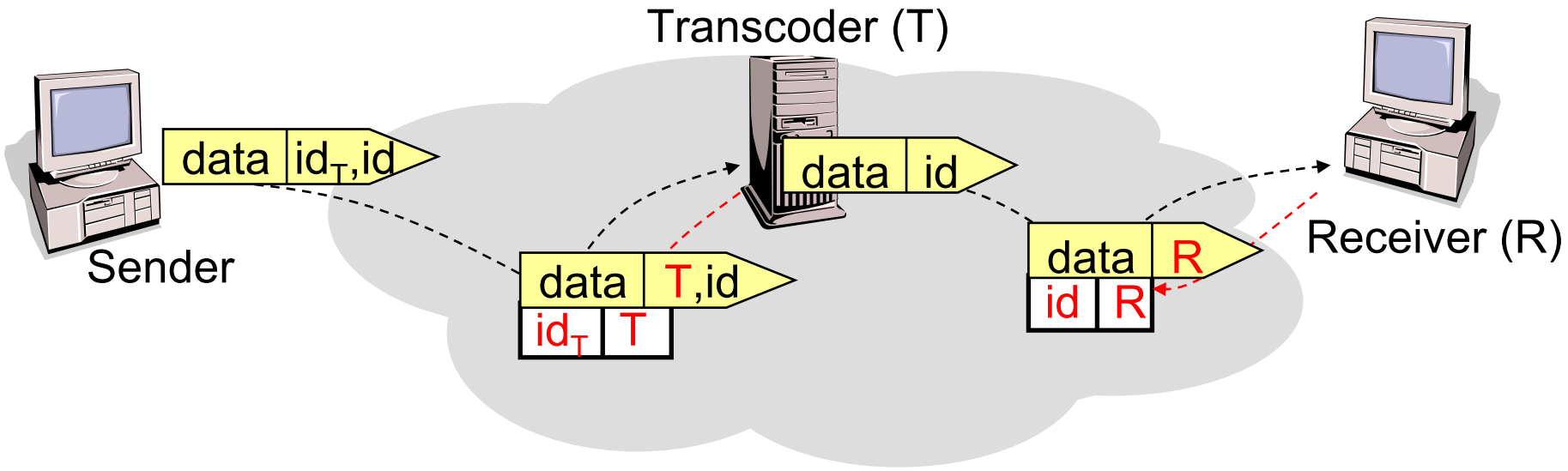
- i3 nutzt ein Routing-Overlay (Chord) um ID-Tupel im Netz zu platzieren
- Nachrichten tragen entweder die Adresse eines Rechners oder den Teil eines Tupels.
- Treffen solche Nachrichten auf ein Tupel, wird die ID gegen die (weiteren) ID bzw. Adressen aus dem Tupel ausgetauscht.
- Beispiele:
  - Mobilitätsunterstützung – Der „Mobile Node“ ist unter einer konstanten ID erreichbar und hinterlegt seine „foreign address“ mittels eines solchen Tupels im Netz.
  - Multicast – Empfänger tragen sich im Tupel der entsprechenden Gruppe ein. Bei großen Gruppen werden mehrere Indirektionen verknüpft.

i3 Anycast



- Das i3 Schema ist mächtig, weil es eine Vielzahl verteilten Anwendungen unterstützen kann.
- Beispielsweise mittels Anycast, d.h. Nachrichten gelangen zu einem (besonders geeigneten) Empfänger.
- Dabei werden nicht vollständige IDs verglichen, sondern ein Longest-Prefix-Matching durchgeführt.
- Anycast ist nützlich z.B. für verteilte Verzeichnisdienste.

i3 Service Composition



- Enthalten Nachrichten mehrere IDs, können die Nachrichten vor ihrer Auslieferung von anderen Diensten bearbeitet werden.
- Diese Bearbeitungsstufen können wieder durch Trigger dynamisch festgelegt werden.
- Beispiel: Transcodierung von Multimedia-Daten für spezifische Empfänger.

Skype Revealed – Not quite …

Über Skype

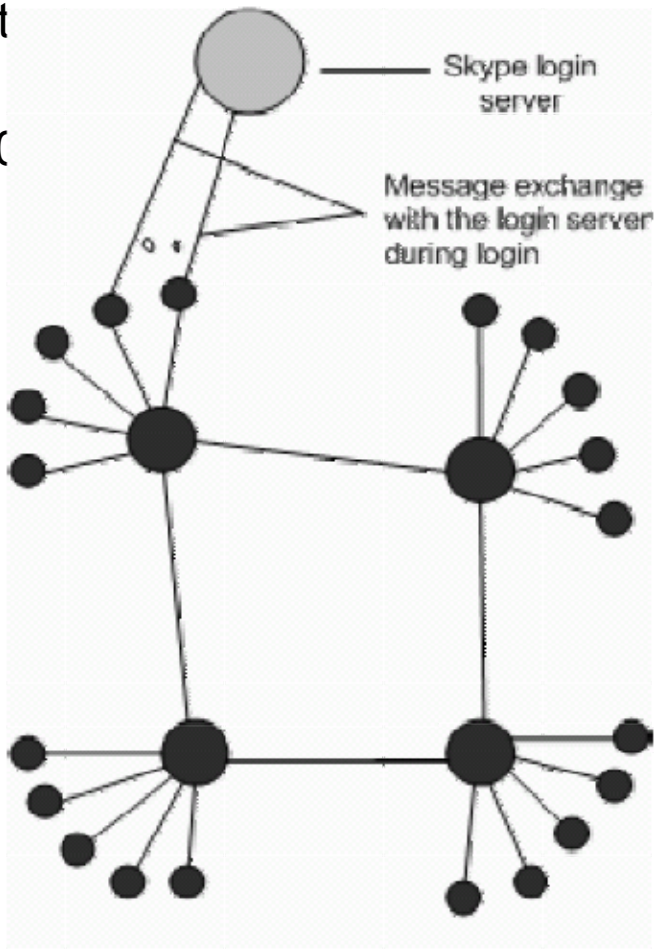
- Skype ist die wohl verbreitetste Internet-“Telephonie“ Anwendung mit zurzeit 42.4 Mio. Nutzern pro Tag
- Telefonate zwischen Teilnehmern sind kostenlos
- Kostenpflichtige Mehrwehrtdienste bieten
  - Festnetztelephonate (Skype-Out)
  - Telefonnummer (Skype-In)
  - Voicemail
  - Voicemail to Text
  - SMS
- Von den gleichen Entwicklern wie Kazaa
- Benützt wie Kazaa eine Superpeer Architektur
- Skype betreibt einen extremen Aufwand um die interne Arbeitsweise zu verschleiern
- Es gibt bis auf den Logindienst keine Zentralen Server
- Komplette online / offline Userinformationen liegen im Netz

Skype Verschleierungstechniken

- Skype executable ist teilweise verschlüsselt und gepackt
- Benützt intern teilweise Code-Checksummen um Sprungziele zu bestimmen
- Durch die Checksummen funktionieren Software-Breakpoints nicht
- Skype misst Timer um zu bestimmen, ob ein Debugger am Skype-Prozess hängt
- Es wird getestet ob bestimmte Debugging-Software (wie Softice) läuft
- Wenn Skype einen Debugger bemerkt, zerstört es die CPU-Register, Stackframe, und springt an eine zufällige Adresse.

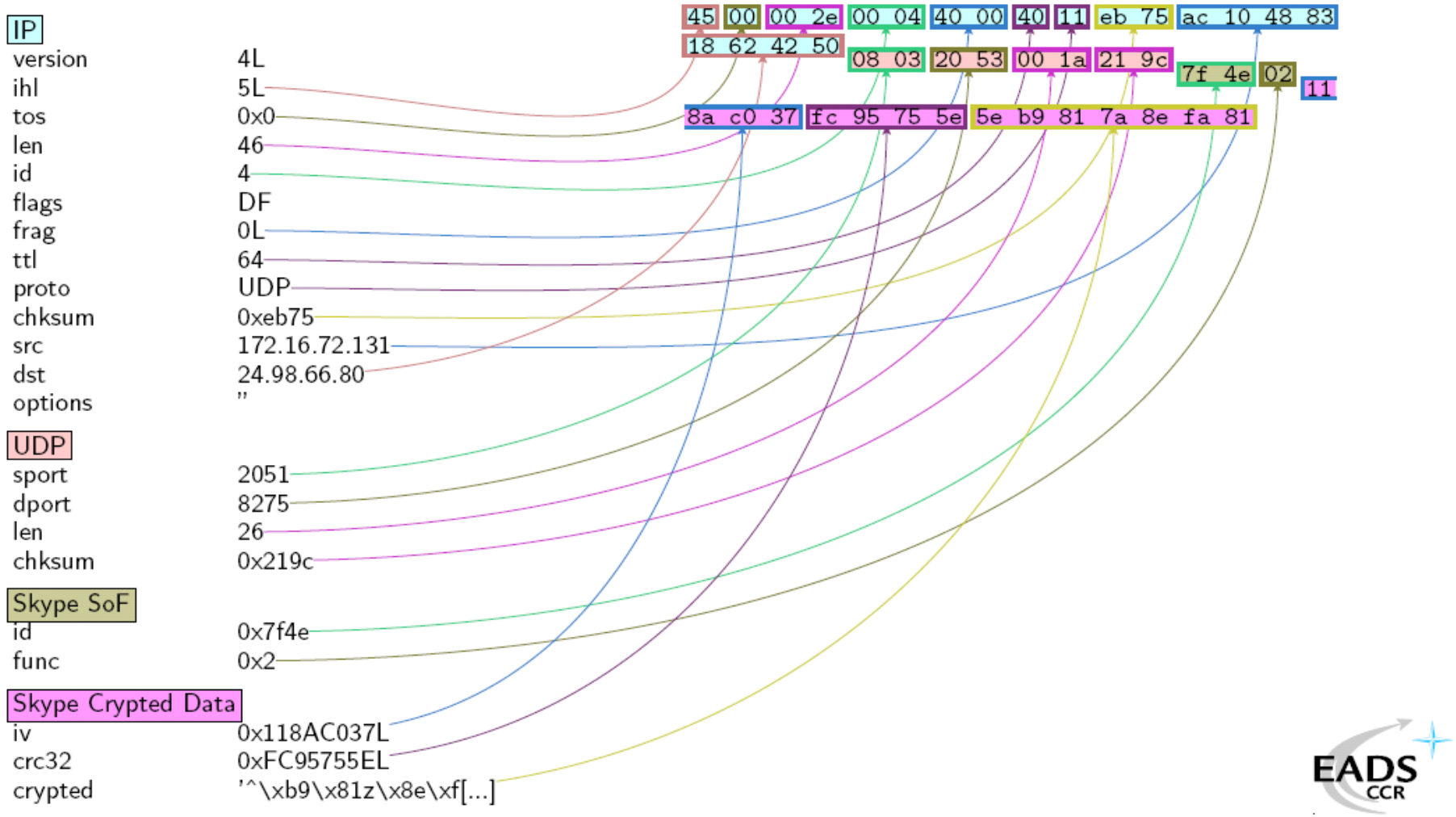
Skype Architektur

- Supernodes sind Rechner mit großer Bandbreite, keiner Firewall und guter CPU
- Jeder Client hält zum Boootstrappen eine List mit Adressen von max. 200 Supernodes
- Insgesamt ca. 20.000 Supernodes (Stand 20(
- Supernodes kennen fast alle anderen Supernodes
- Skype ist problematisch für Netzwerkadministratoren:
  - Verkehr verschlüsselt
  - Keine festen Portnummern
  - Verwendet TCP/UDP, u.A. auf Port 80 und 443
  - Kann auch Proxies verwenden
  - Skype hat Application to Application Protokoll, so dass es als Proxy für jede Art von Anwendung dienen kann



Bildquelle: Baset, S. and Schulzrinne H.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol; Proc. Infocomm 06

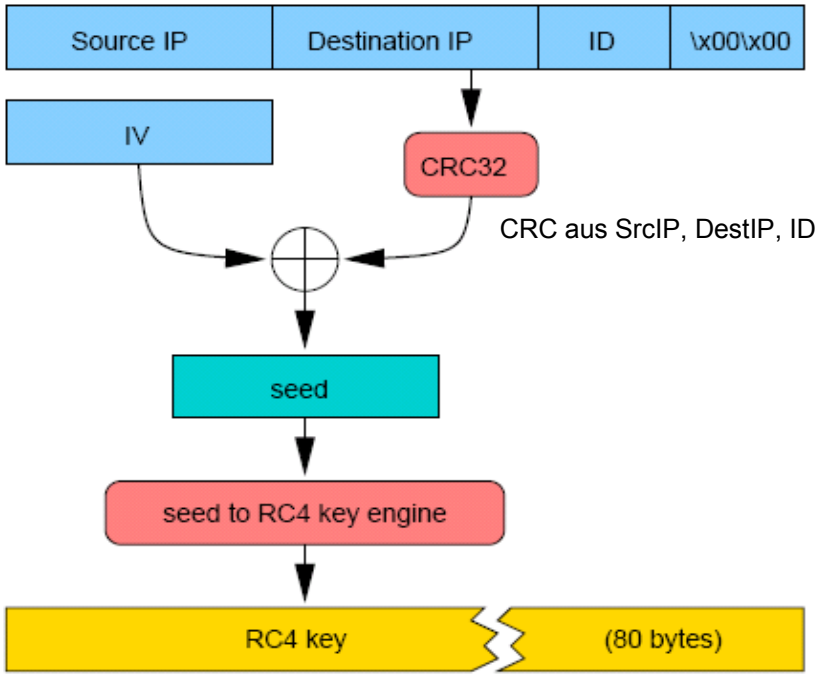
Skype UDP Network Pakete



Bildquelle: Philippe Biondi, Fabrice Deslaux: Silver Needle in the Skype. BlackHat Europe, 2006

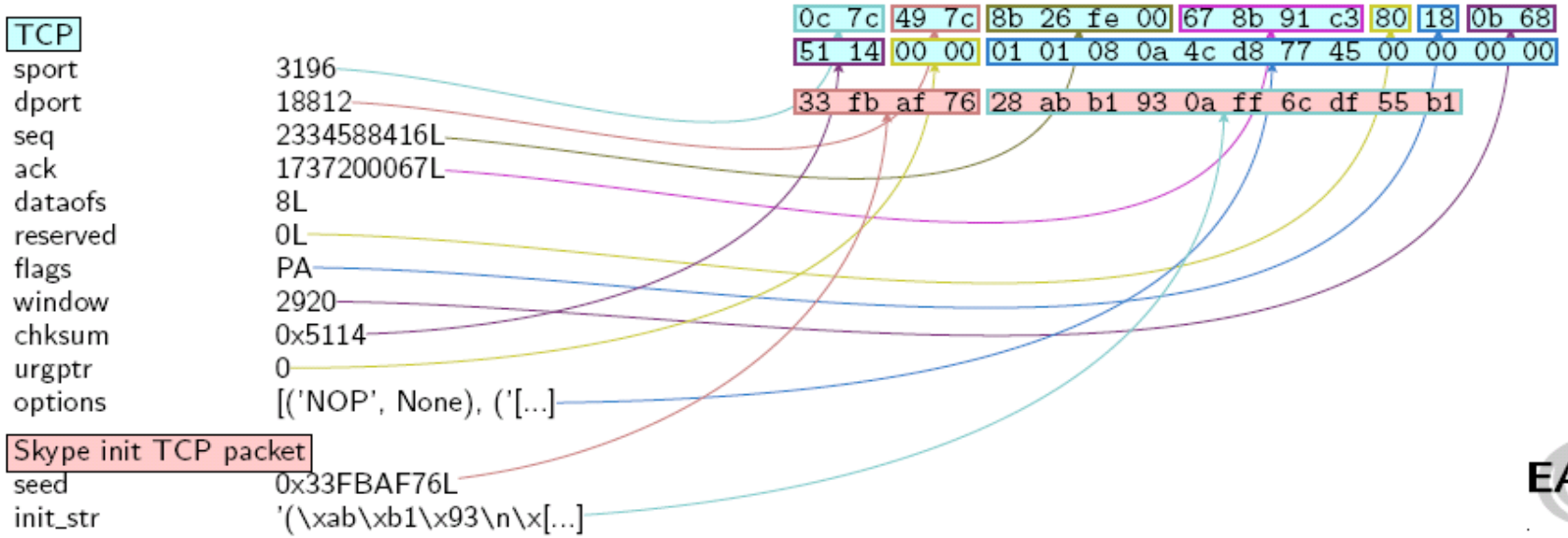
Skype UDP Paketverschleierung

- Fast alle Pakete, die zwischen zwei Skype-Hosts ausgetauscht werden sind mit RC4 verschlüsselt
- Sprachpakete benützen eine AES Verschlüsslung
- Bei RC4 wird der Seed der Key-Engine berechnet aus
  - Source IP
  - Destination IP
  - Packet ID
  - Initialization Vector
- Um die öffentliche IP-Adresse herauszufinden senden Skype-Hosts ein UDP-Paket, in der als Source-IP 0.0.0.0 eingetragen ist
  - Zielrechner kann Nachricht nicht entschlüsseln und sendet Klartext NAck mit IP des Senders zurück



Bildquelle: Philippe Biondi, et al.: Silver Needle in the Skype. BlackHat Europe, 2006

Skype TCP Network Pakete



Bildquelle: Philippe Biondi, Fabrice Deslaux: Silver Needle in the Skype. BlackHat Europe, 2006

Skype Blockieren

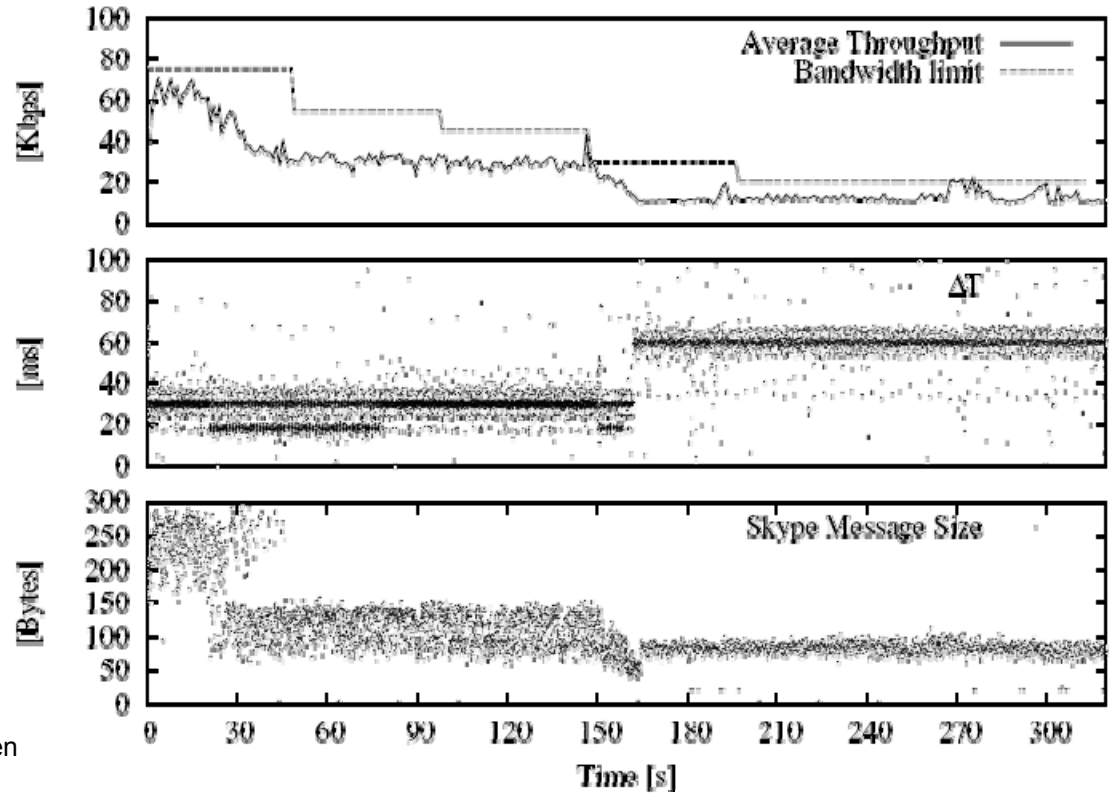
- Man kann das öffentliche NAck von Skype auf UDP-Pakete blocken. Damit ist kein Skype UDP-Verkehr mehr möglich.

- Das Blockieren des Skype TCP-Verkehrs ist schwerer, da hier als erstes ein Random-Seed geschickt wird, nach dem nur noch verschlüsselter Verkehr folgt.
- Zwei Lösungen
  - Falsche Antworten auf Skype UDP-Anfragen schicken,

Skype Traffic Identification

- Zwei Verschiedene Modi in denen Skype arbeiten kann
  - End-to-End zwischen zwei Skype-Instanzen
  - End-to-Out für Verbindungen ins Telefonnetz (Skype-Out)
- Skype benutzt verschiedene Audio-Codecs mit bekannten Eigenschaften.
- Diese Eigenschaften kann man nutzen, um Skype-Traffic im Netz zu erkennen.

Codec	Frame Size [ms]	Bitrate [Kbps]
ISAC	30,60	10 ÷ 32
ILBC	20,30	13.3, 15.2
G.729	10	8
iPCM-wb	10,20,30,40	80 (mean)
EG.711A/U	10,20,30,40	48,56,64
PCM A/U	10,20,30,40	64



Bildquelle: Bonifiglio et. Al: Revealing Skype Traffic: When Randomness Plays with You. SIGCOMM 2007

Eigene Arbeiten zum Thema Peer-to-Peer

Übersicht

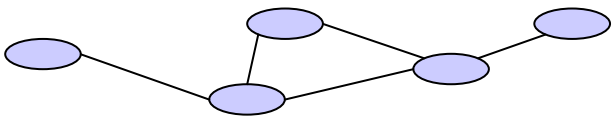
Peer-to-Peer

Videgor
VDR Plugin
verteilter
Videorekord.

IgorFS
Verteiltes
Dateisystem

...

Igor Overlay



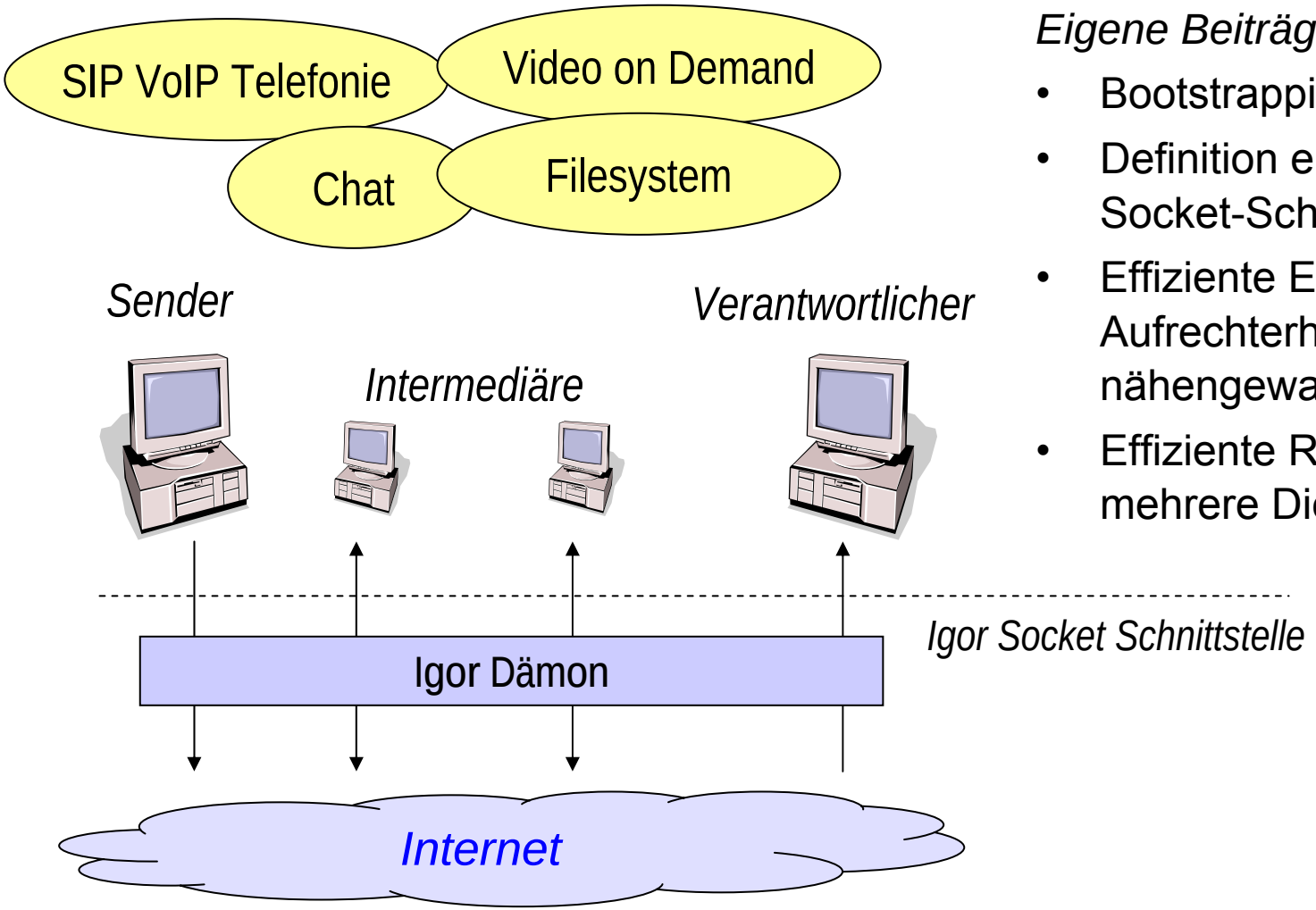
Verteiltes Rechnen

Ambicomp
Sensor-Aktornetze
in Java

JCell
Manycore
Hochleistungsrechnen

SSR

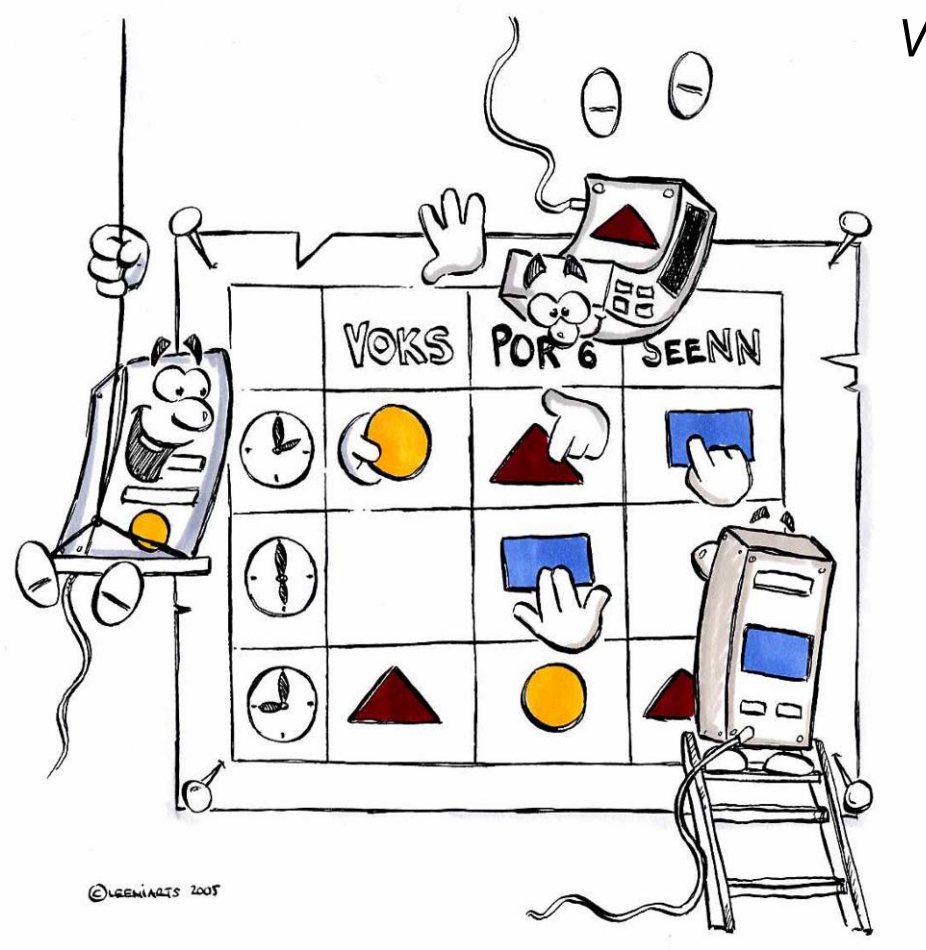
Igor als P2P-Service-Middleware



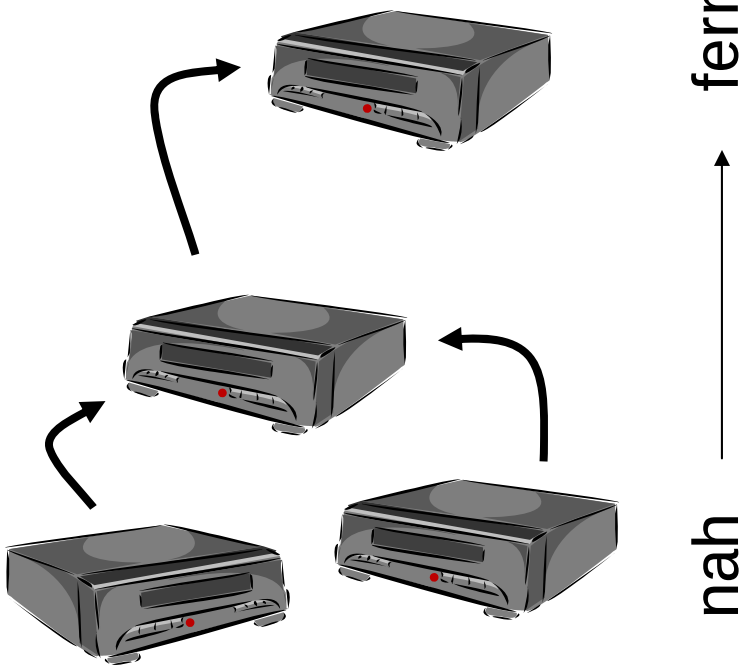
Eigene Beiträge:

- Bootstrapping P2P-Netze
- Definition erweiterter Socket-Schnittstelle
- Effiziente Etablierung und Aufrechterhaltung nähengewahrer Overlays
- Effiziente Realisierung mehrere Dienste

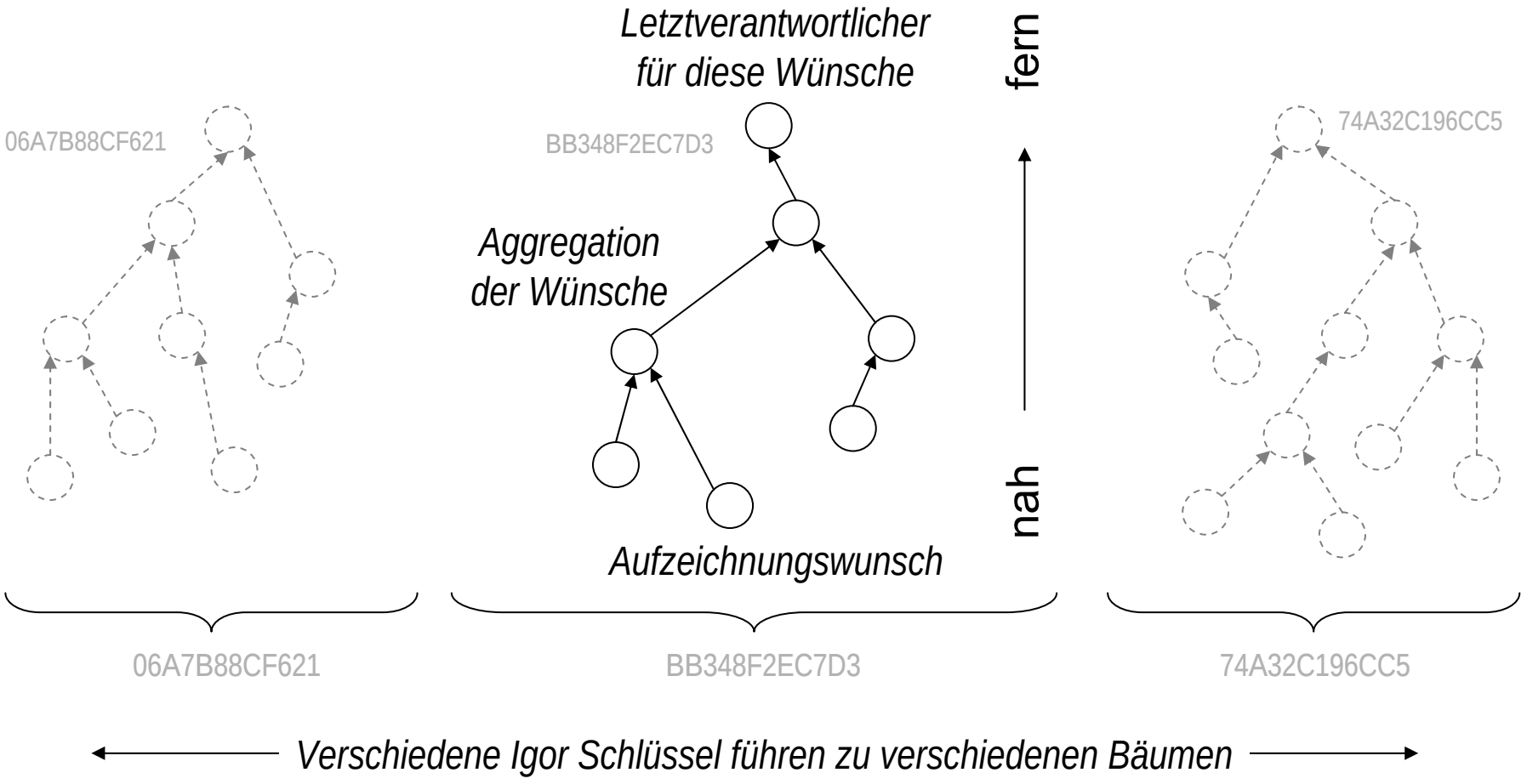
Peer-to-Peer Videorekorder „Videgor“



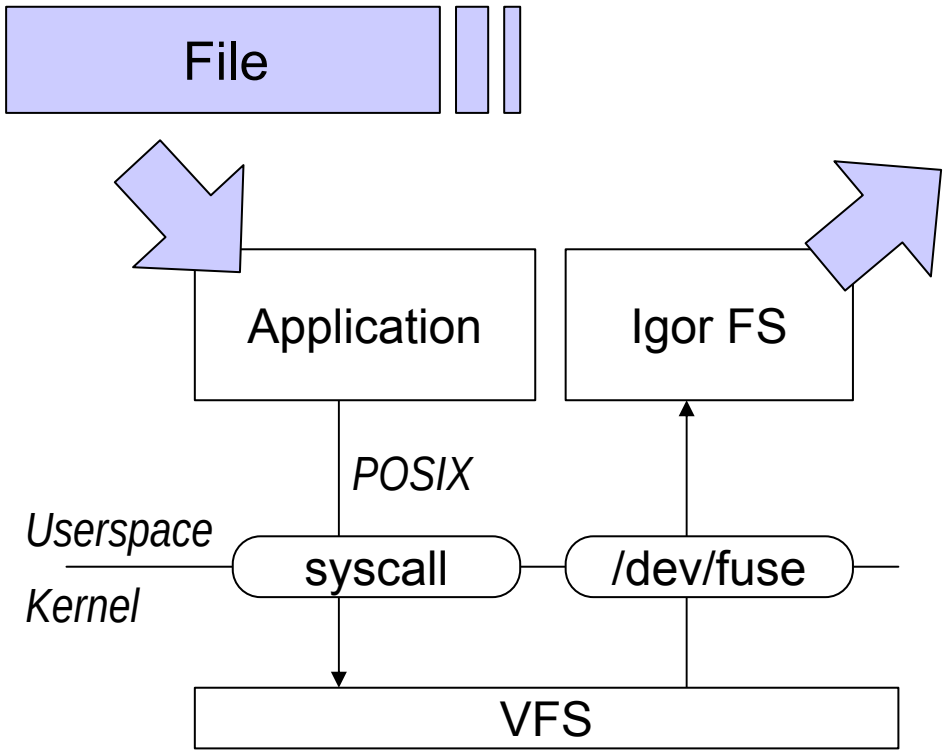
Videgor: Dezentrales Scheduling & Zugriff auf Videodateien

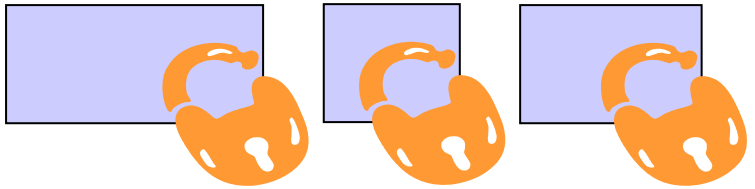


Videgor – Ein robuster dezentraler Dienst

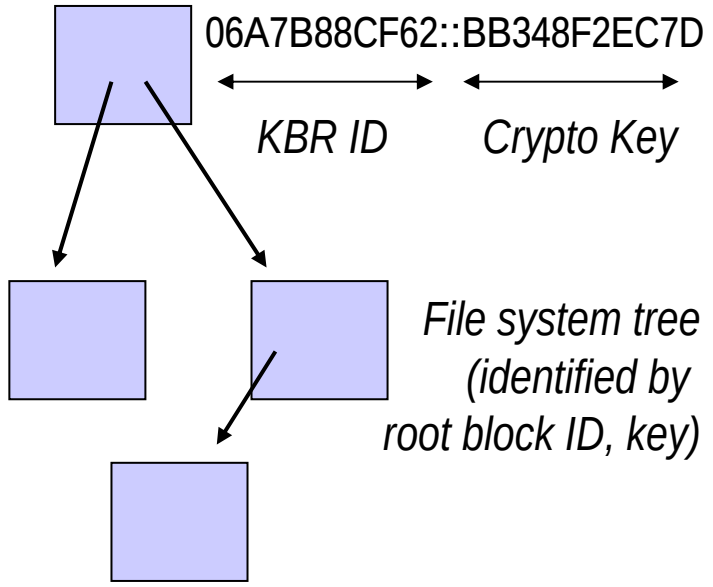


Peer-to-Peer Filesystem „Igor-FS“



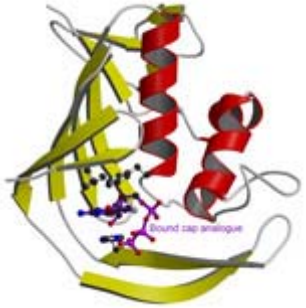


Variable sized data chunks



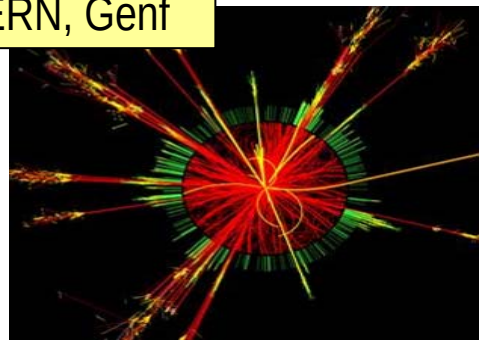
Anwendungsgebiete von Igor FS

EMBL, Heidelberg



- Große Dateien in der Bioinformatik
- Mehrere zentrale Datenquellen (für getrennte Verzeichnisse)
- Häufige, kleine Änderungen
- Viele Abonnenten, die jeweils nur kleine Abschnitte der Dateien lesen

CERN, Genf



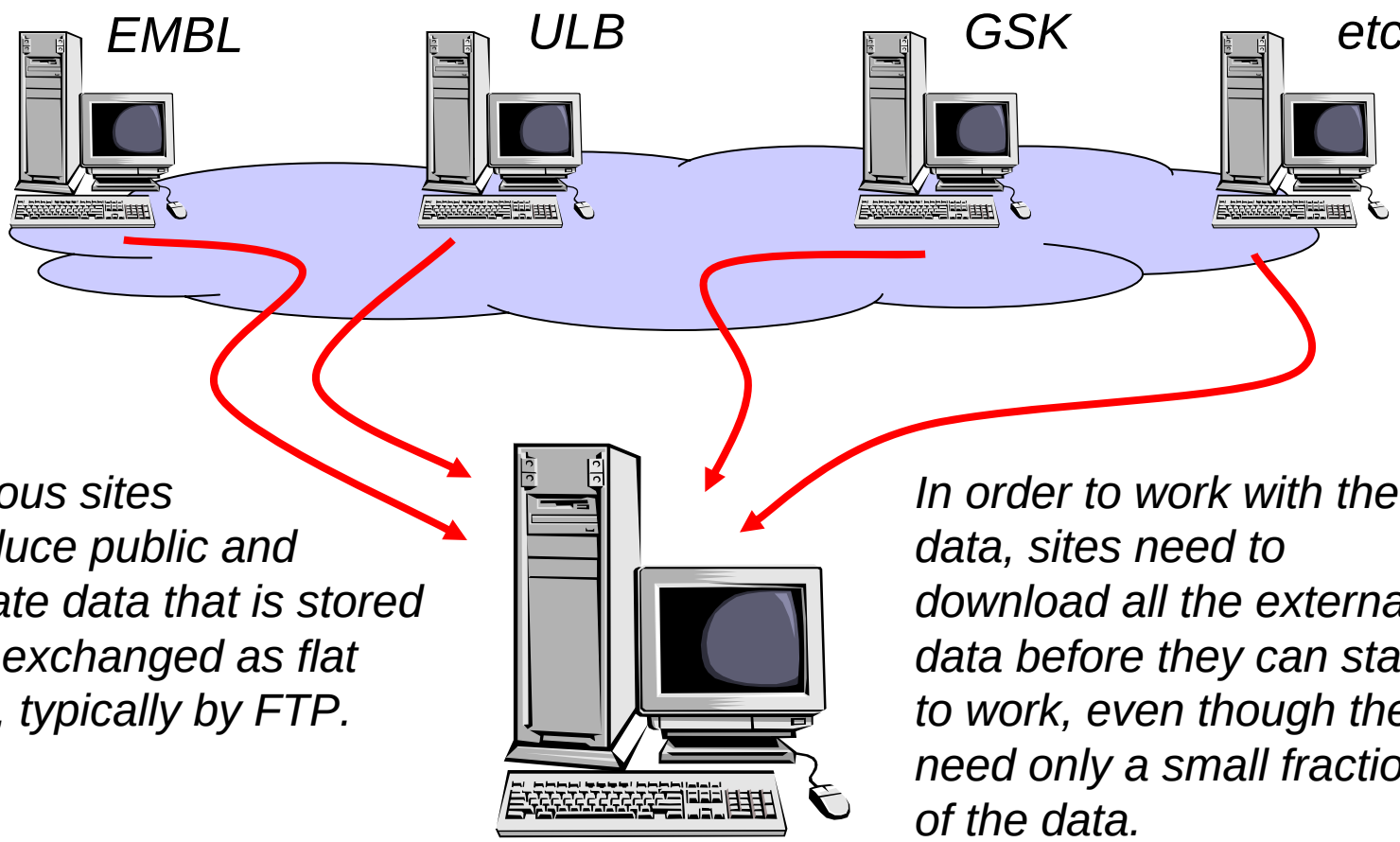
- Große Zahl von Konfigurationen (OS + Bibliotheken + Analyse-Software)
- Große Zahl von Rechnern im Cluster
- Häufige Konfigurationswechsel, Cold Standby, etc.
- Robust & Effizient → Dezentralität

- Verteiltes Dateisystem für allgemeine Anwendungen
- Benutzer, Gruppen
- Konflikte?
- Fairness?

Aktuelle Arbeiten

Kontakt:
Bernhard Amann
Tel. -18035

The EU FP6 SIMDAT Pharma Challenge

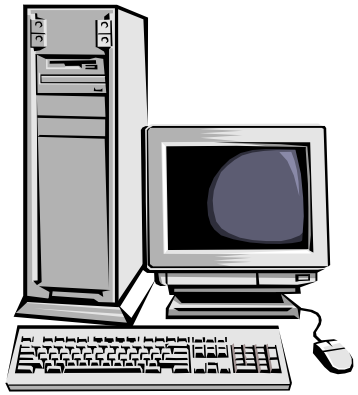


Various sites produce public and private data that is stored and exchanged as flat files, typically by FTP.

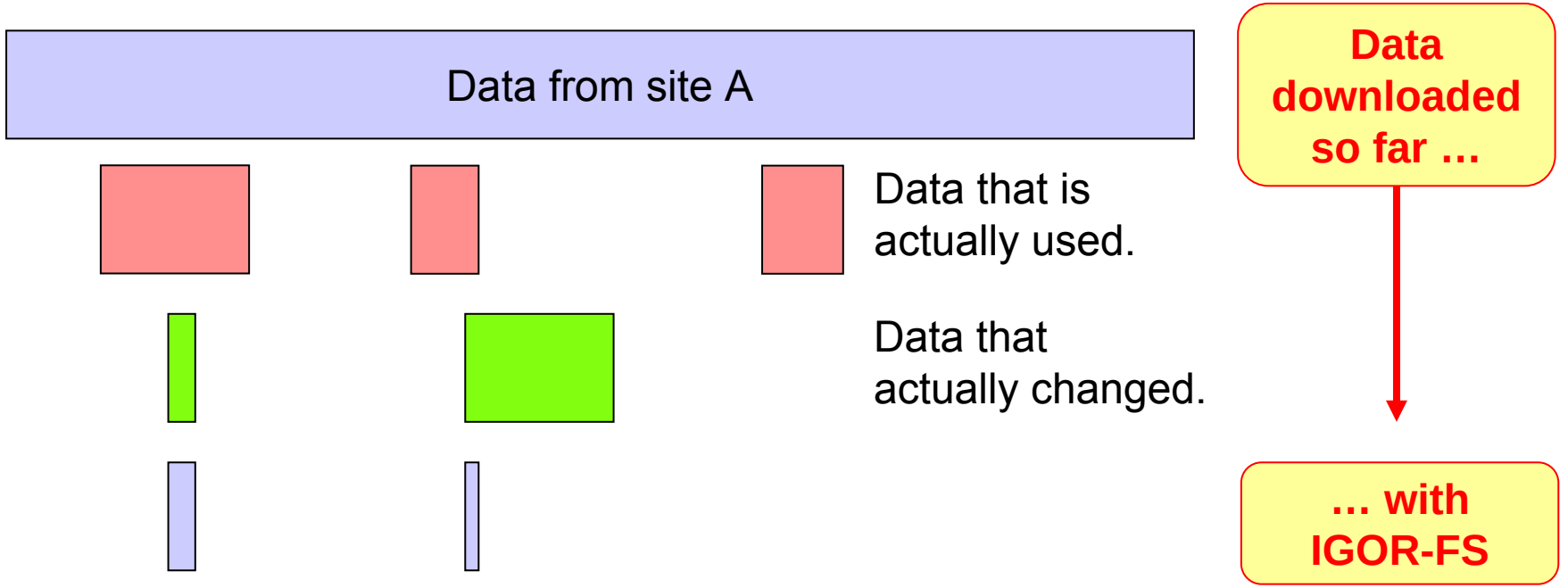
In order to work with the data, sites need to download all the external data before they can start to work, even though they need only a small fraction of the data.

The IGOR-FS Solution (1)

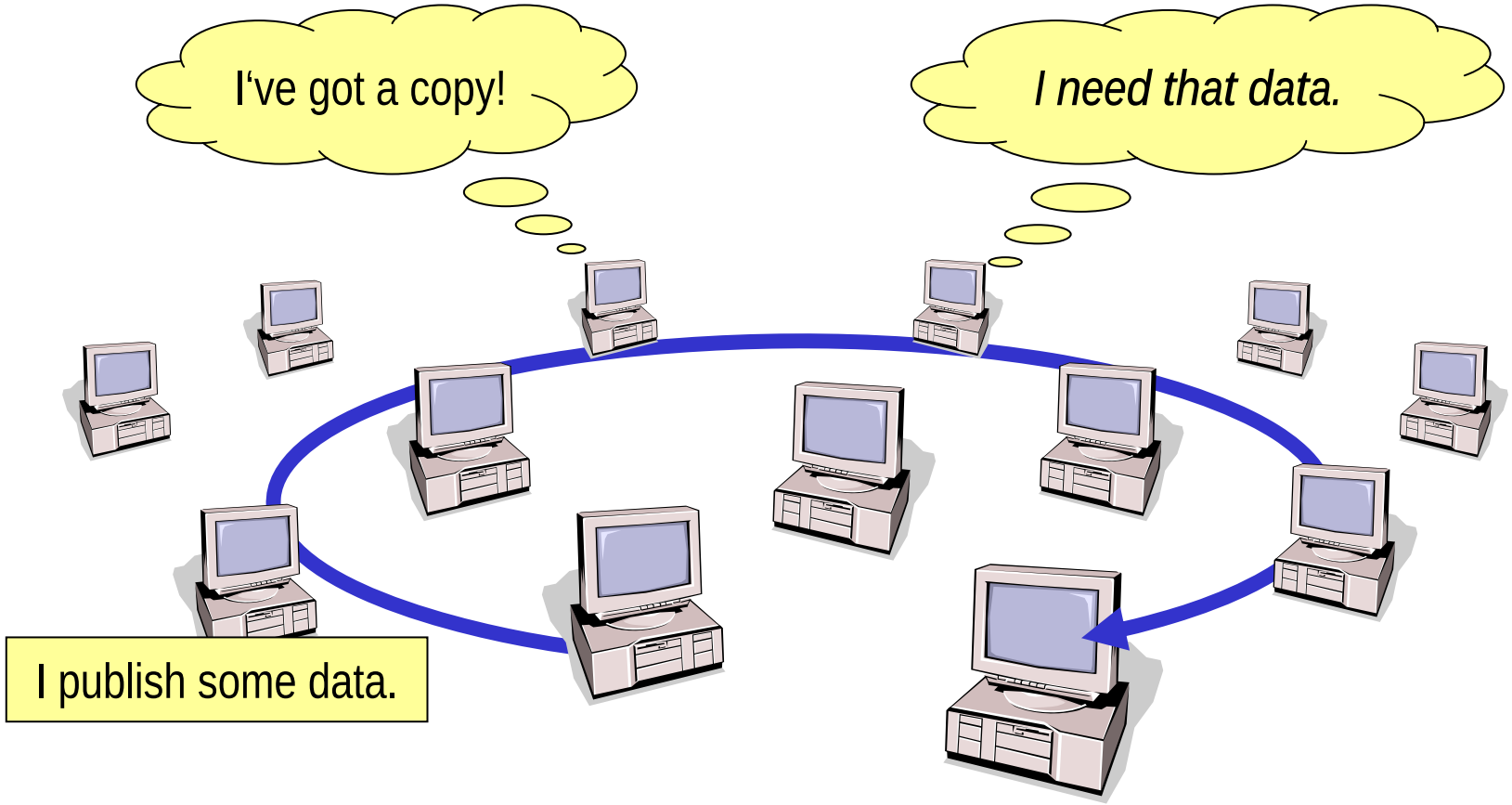
Today, sites use only a few sources in practice. Ideally, their system could handle cross-references between these sites.



But in fact they use only a small part of that data, and they reload the whole data even though it did not change much.



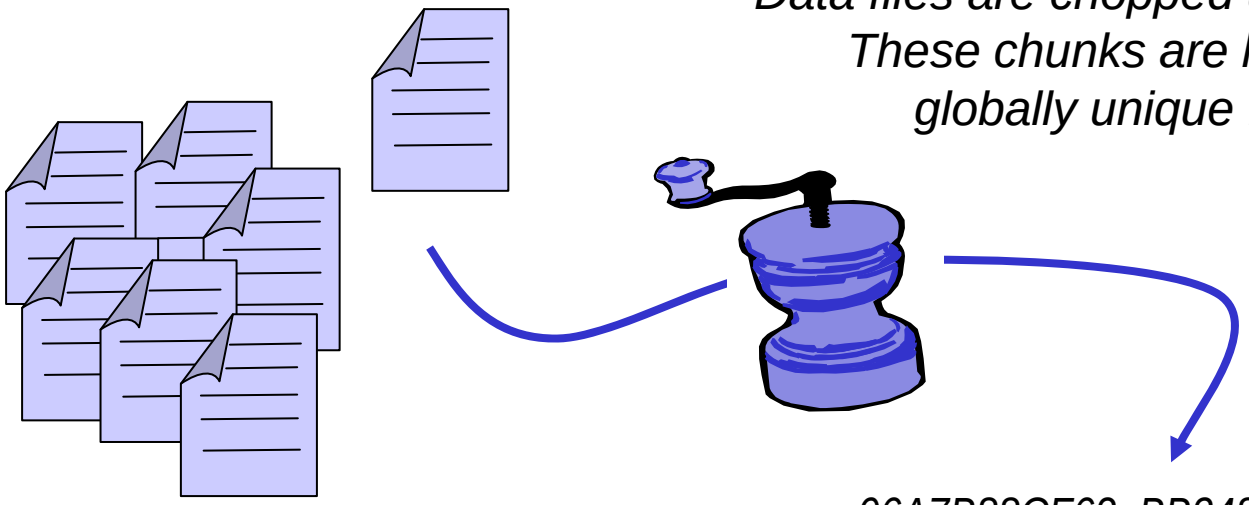
The IGOR-FS Solution (2)



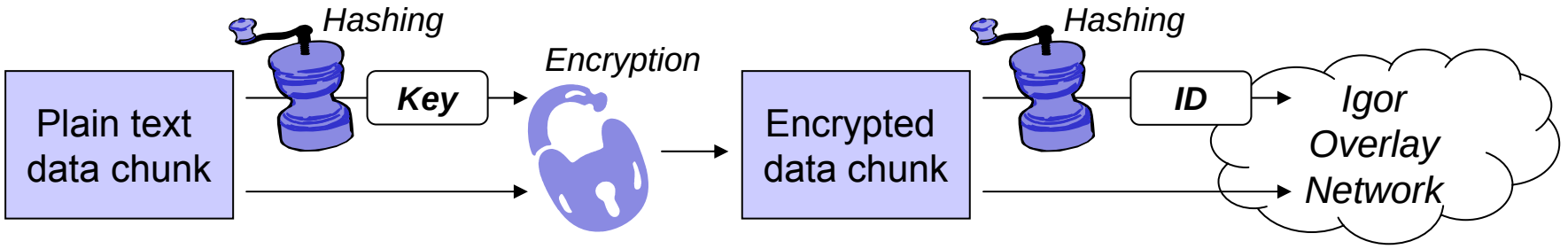
IGOR-FS shares the data that is published and used in the community in a peer-to-peer-like manner.

Sharing Files in IGOR-FS

Data files are chopped up in chunks. These chunks are hashed into a globally unique ID and a key.



06A7B88CF62::BB348F2EC7D



Chopping up Files in IGOR-FS

neque porro quisquam est, qui do-
lorem ipsum
quia
dolor sit, amet, consectetur,
adipisci velit, sed
quia non numquam
eius modi tempora incidunt, ut
labore et dolore magnam aliquam
qua
erat
voluptatem.

Cicero, „De finibus bonorum et malorum“

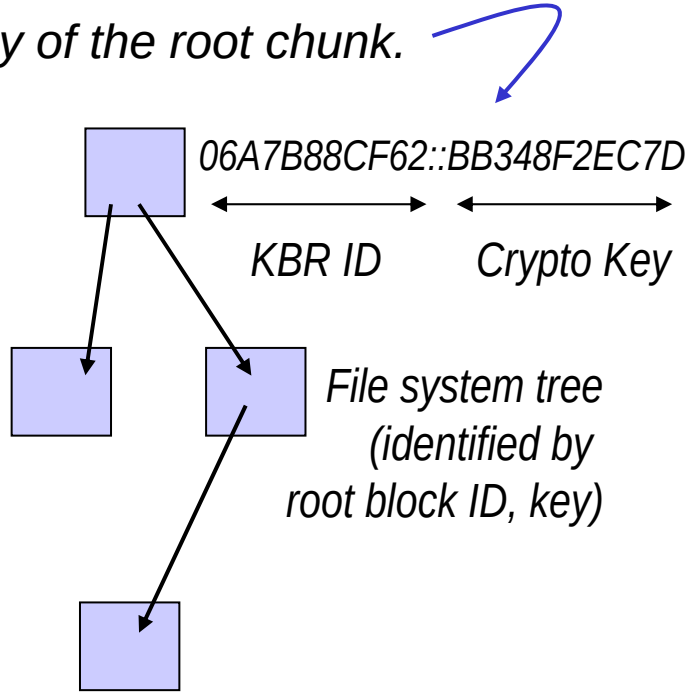
lorem ipsum
dolor sit amet, consectetuer
adipsci velit, sed
diam nonumy
eius modi tempora incidunt, ut
labore et dolore magnam aliquam
erat
sed diam
voluptatem.

A common placeholder text

Note: Chunks in IGOR-FS have ca. 50-100 KiB. They are cut at well-known boundaries or by using a rolling checksum.

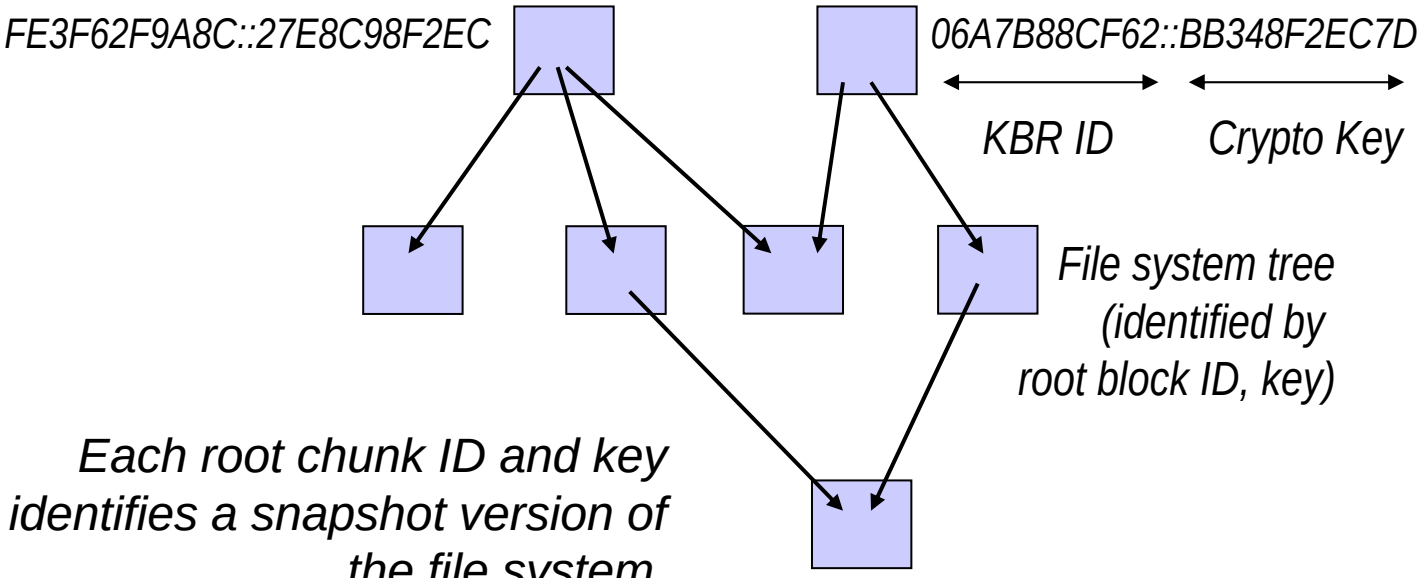
Backup & Versioning (1)

1. *Obtain ID and key of the root chunk.*



2. *Root chunk then contains IDs and keys of the files (and folders) of the file system's root level, and so on …*

Backup & Versioning (2)

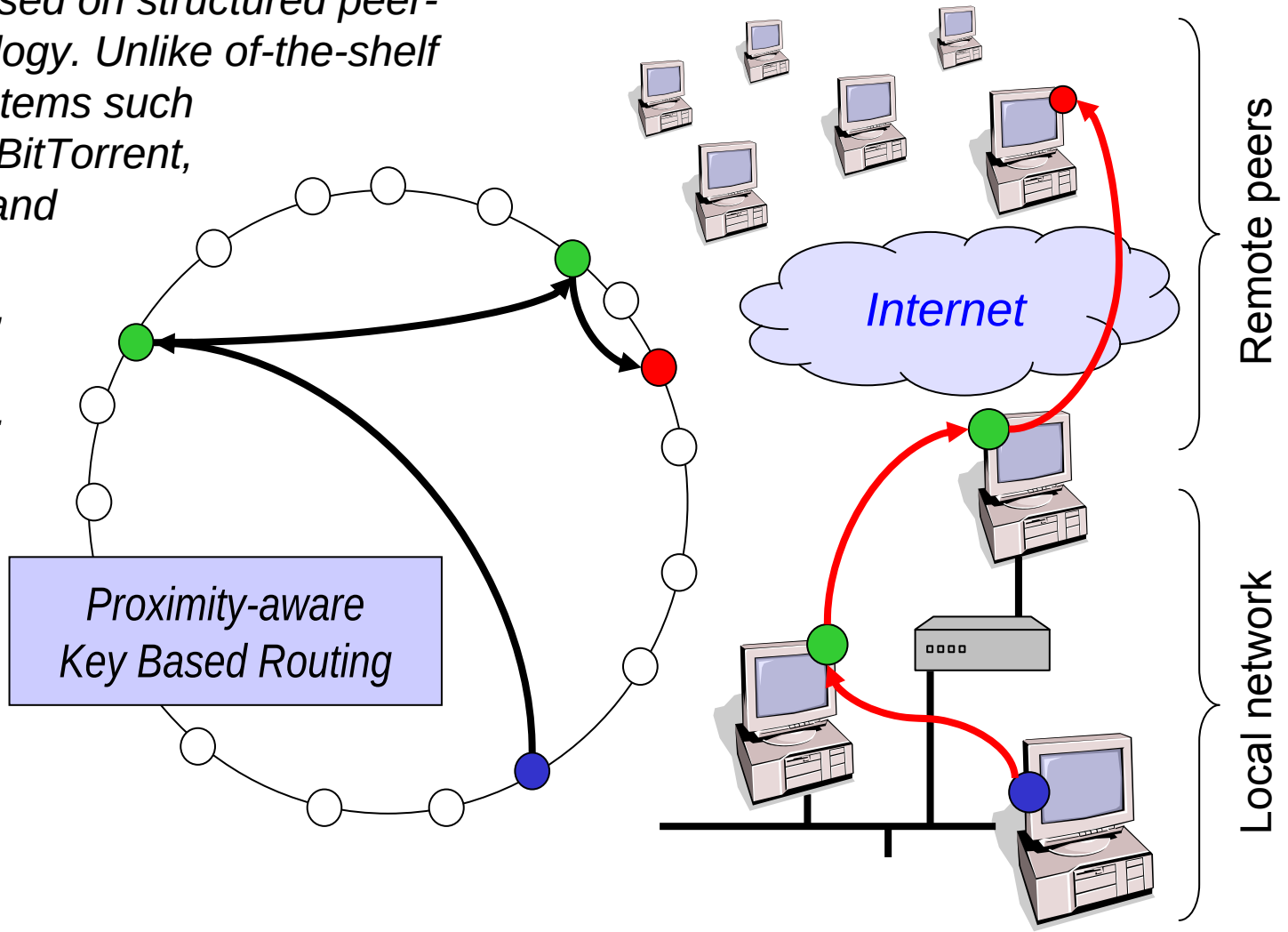


Each root chunk ID and key identifies a snapshot version of the file system.

New chunks only reflect modifications in content and structure.

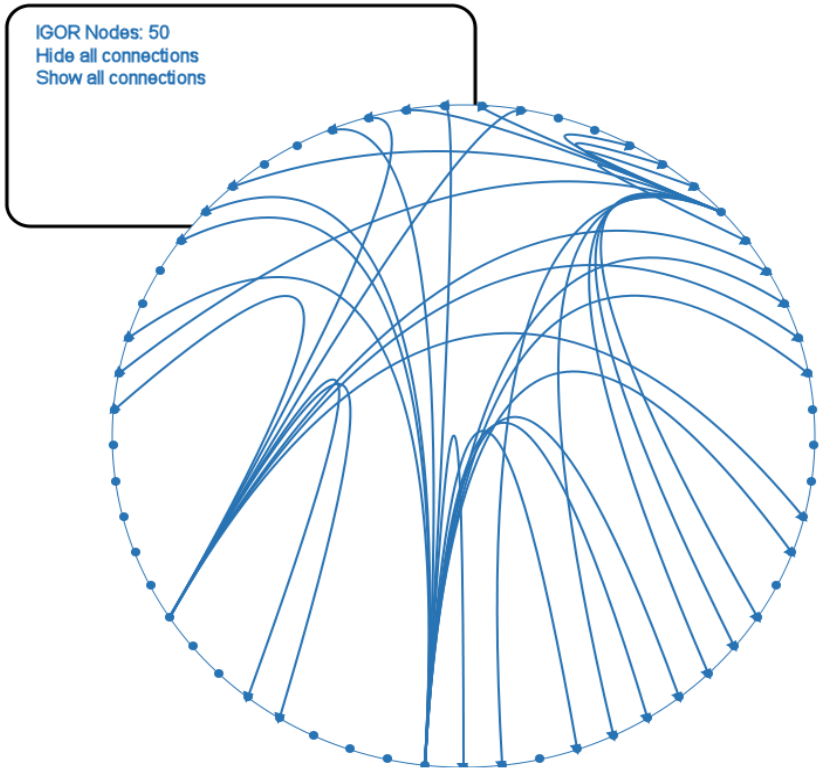
Finding Data in IGOR-FS

IGOR-FS is based on structured peer-to-peer technology. Unlike of-the-shelf file sharing systems such as Gnutella or BitTorrent, data is stored and retrieved in a self-organizing, but highly structured way.



The IGOR Daemon

- IGOR = Internet Grid Overlay Routing (or Implementing the Grid with Overlay Routing), but actually an allusion to Terry Pratchett's *Disc World* novels
- IGOR has bee written by our group as a research platform (cf. PhD thesis of Kendy Kutzner 2008, now with Google in Zurich)
- Standard UNIX daemon
  - /etc config dir
  - init scripts
  - privilege drops
- TCP connections
  - RTT measurements for free
  - Retransmit, Reliability, Fault Tolerance
- Powerful Plugin Concept
- Currently runs on about 300 PlanetLab nodes



IGOR Plugins

Plugin Interface

cMessagePlugin

Register new message types

cPolicyPlugin

Routing table policies, e.g.

- ConnectionOpened()
- ConnectionClosed()
- EvalForRouting()
- EvalForConnectionDrop()
- EvalPotentialConnection()

cPeekPlugin

Look at incoming and outgoing messages

cTaskPlugin

Do something at a regular basis

Plugins are used for …

Limitation Policy:

Limit the number of Connections

Fix Fingers Task:

Obey the constraints of the underlying routing geometry

Dump Cache:

Write active connections to a file and bootstrap from it

Service Task:

Provide not only a specific service, provide *n* services

NAT:

Try to handle NAT

SSL:

Encrypt / Compress Connections

Proximity:

Optimize routing geometry

IGOR's Library Interface

```
libigor:

    igor_socket()
    igor_bind()
    igor_unbind()
    igor_close()
    igor_sendto()
    igor_recvfromto()
    igor_register_callback_neighborset_change()
    igor_get_nodeid()

libdht:

    libdht_put() bzw. libdhq_push()
    libdht_get()
    libdht_erase()
```

Questions?

Thomas Fuhrmann

Department of Informatics
Self-Organizing Systems Group
c/o I8 Network Architectures and Services
Technical University Munich, Germany

fuhrmann@net.in.tum.de