

Content-oriented Bridging of Today's Internet Heterogeneity

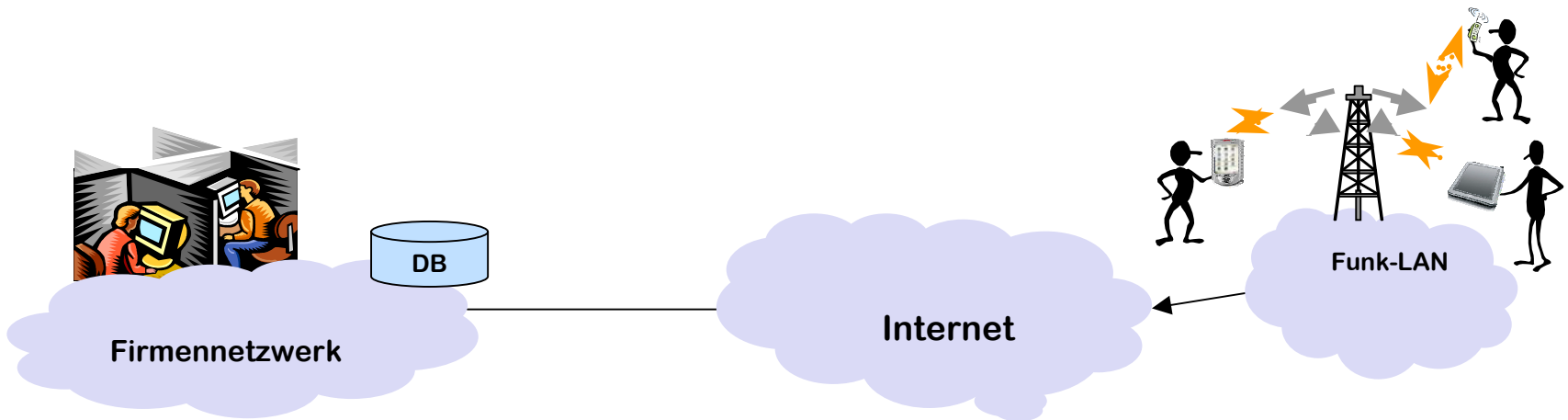
Klaus Wehrle – Stefan Götz

Protokoll-Engineering & Verteilte Systeme
Wilhelm-Schickard-Institut für Informatik – Universität Tübingen

...in Kooperation mit Ion Stoica, University of California at Berkeley

- **Mobile Mitarbeiter einer Firma**

- ▶ Zugriff auf die Dienste und Datenbestände der Firma von überall
- ▶ Erreichbarkeit immer und überall
- ▶ Nutzung der neusten Möglichkeiten heutiger Informationstechnologie
 - Tablet PC, PDA, Smartphone



Probleme und Anforderungen

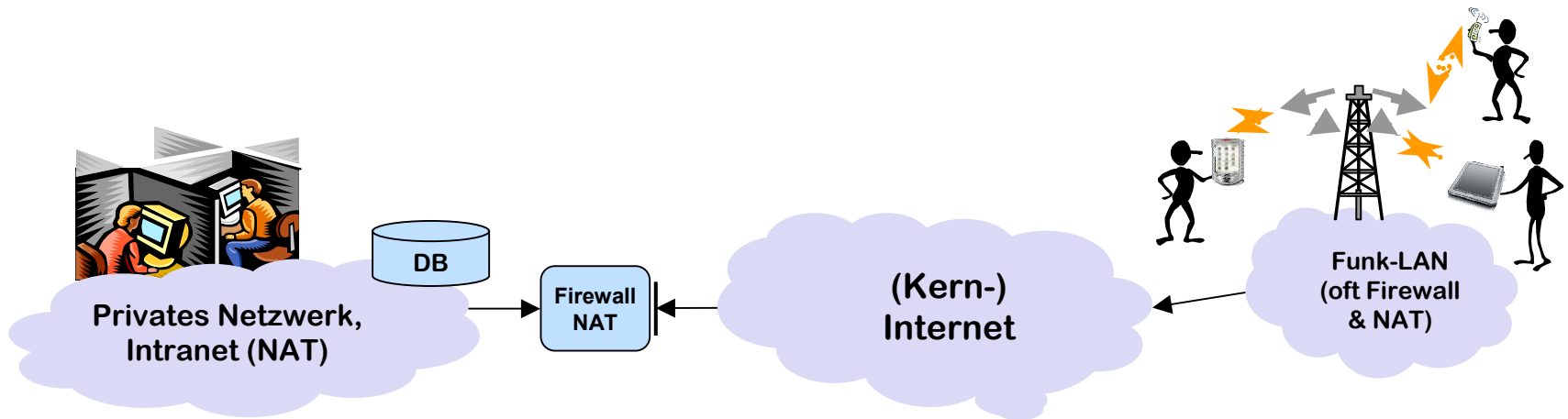
- **Heterogenität**

- ▶ Netzwerk

- Limitierungen durch IP-Architektur
 - Oft Verletzung der IP-Architektur

- ▶ Geräte

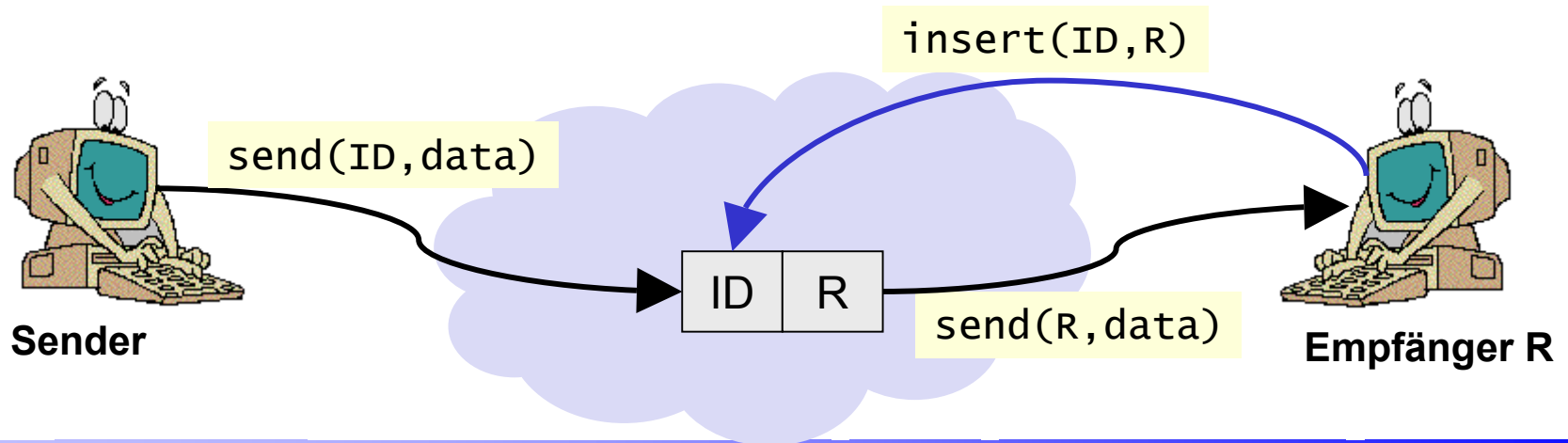
- Darstellungsformate, Rechen- und Speicherkapazität



- **Limitierungen durch IP-Architektur**
 - ▶ IP für Punkt-zu-Punkt-Kommunikation entworfen
 - ▶ Einführung anderer Kommunikationsformen (bislang) wenig erfolgreich
 - Multicast
 - Anycast
 - Mobilitätsunterstützung
 - (aktive Netze)
 - ▶ Adressknappheit
- **Verletzungen der „reinen“ IP-Architektur**
 - ▶ NAT & private Adressräume
 - ▶ Middleboxes

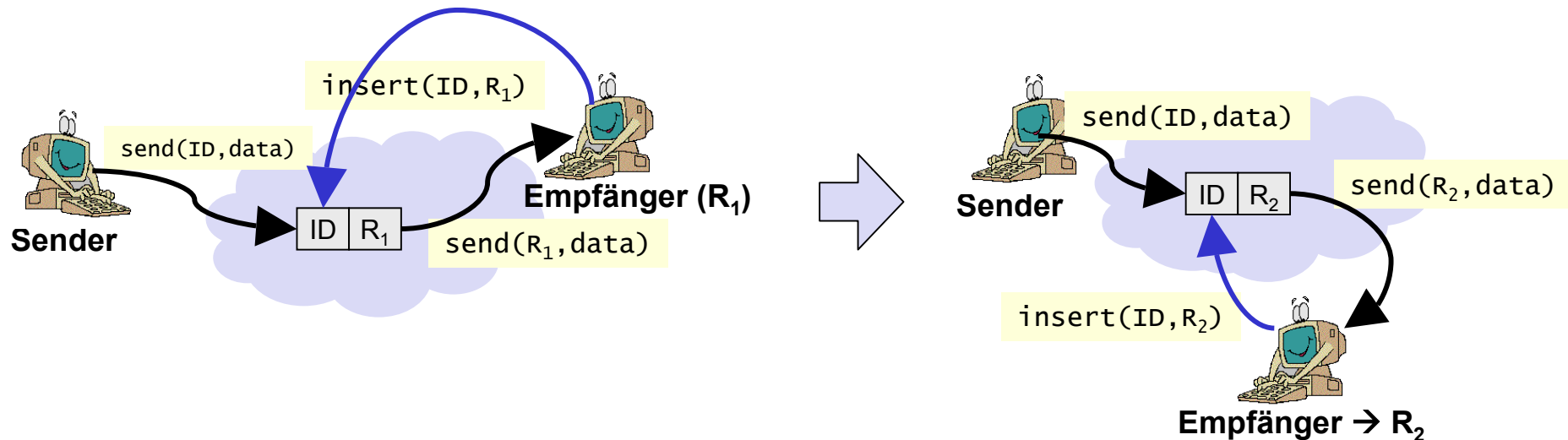
Lösungsansatz: Indirektion

- **Idee:**
Schaffung einer generischen Indirektionsstufe auf P2P-Basis
 - ▶ skalierbar, flexibel, zuverlässig
- **Internet Indirect Infrastructure (i3)**
 - ▶ Empfänger „abonniert“ Inhalt/Dienst (ID) durch Setzen eines Triggers
 - ▶ Sender sendet Daten an ID
 - ▶ Für ID zuständiger i3-Knoten leitet Daten an den/die Empfänger weiter



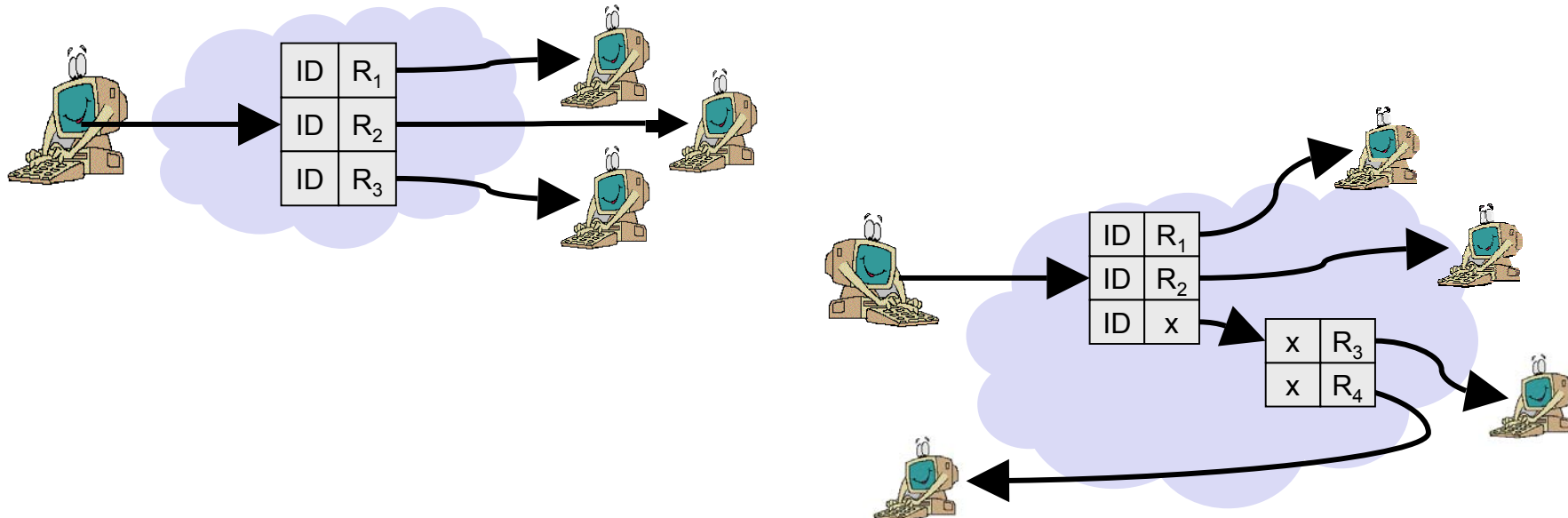
- **Empfänger ändert Zieladresse im Trigger**

- ▶ Pakete werden automatisch zum neuen Standort geleitet
- ▶ Empfänger und Sender können mobil sein
- ▶ Standort nicht ermittelbar (**Location Privacy**)



• Multicast mit i3

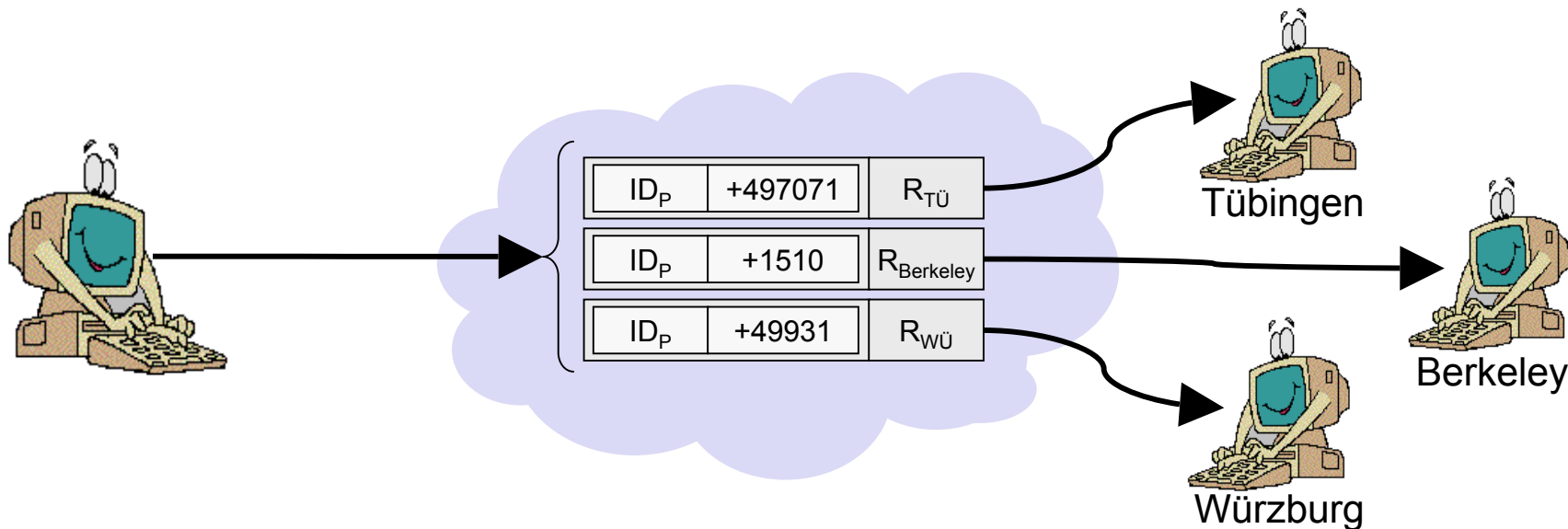
- ▶ Jeder Empfänger setzt einen Trigger
- ▶ Alle Trigger werden im gleichen Knoten verwaltet
- ▶ Skalierbarkeit in großen Gruppen
 - Alle Trigger würden im gleichen Knoten verwaltet werden
 - Lösung: Aufbau eines Verteilbaumes mit weiteren Triggern



Anycast-Kommunikation

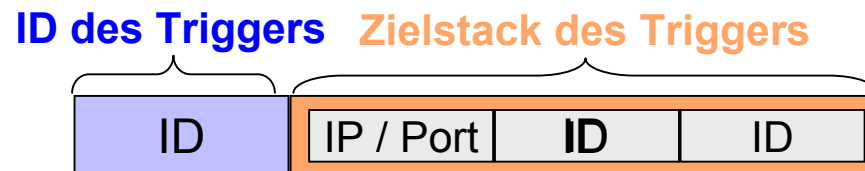
- Anycast mit i3

- ▶ Anycast-Gruppenmitglieder setzen Trigger mit gemeinsamem Präfix P
- ▶ Paket wird an Ziel per “longest prefix match” ausgeliefert
- ▶ Postfix der Mitglieder bzw. des Pakets bietet interessante Möglichkeiten
 - Load-Balancing, Proximity, Random



Verkettung von Diensten mit i3

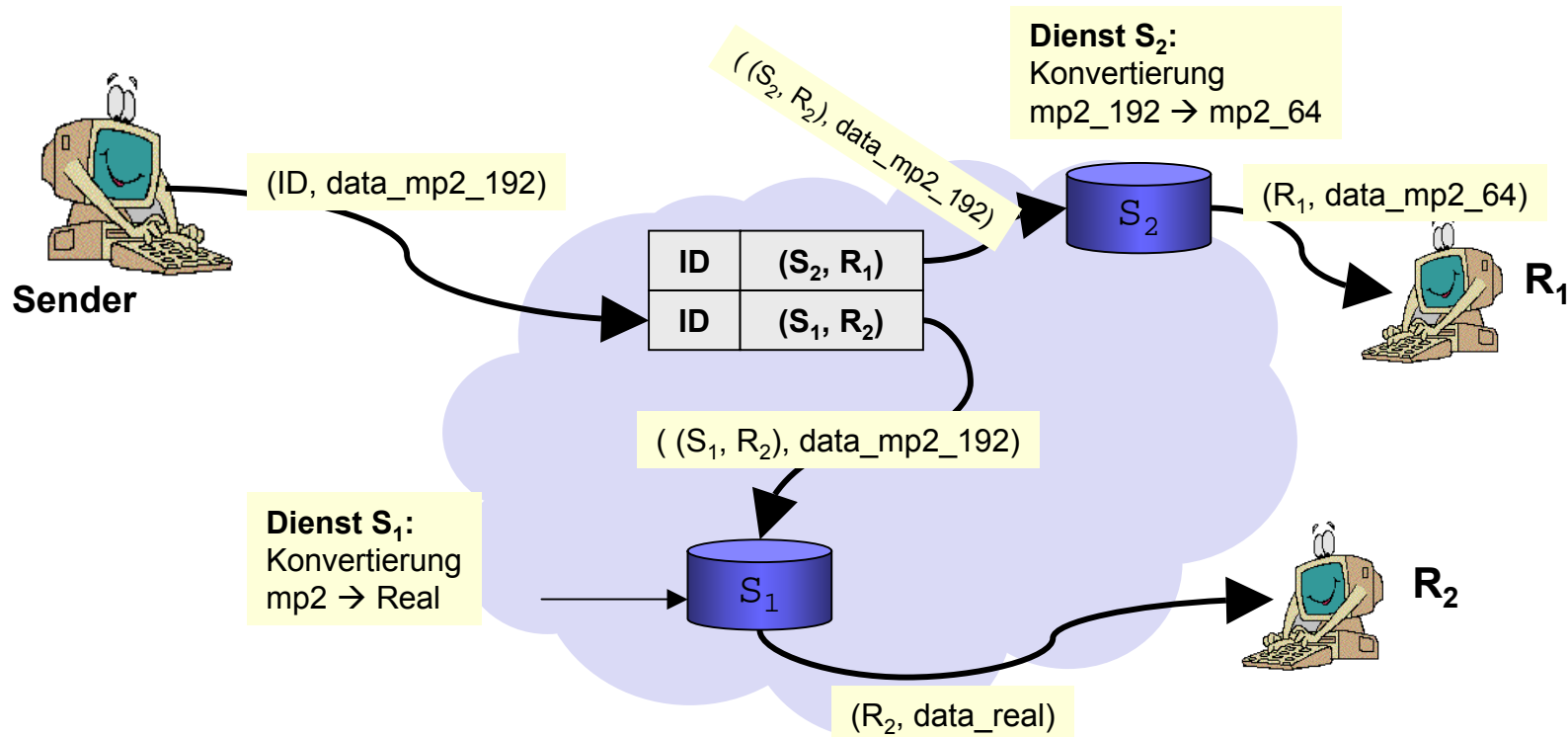
- **Datenstrom soll im Netz mehrere Dienste erfahren**
 - ▶ Ansatz aus dem Bereich der aktiven Netze
 - ▶ Beispiel:
 - (Um-)Kodierung von Inhalten, z.B. mp2 → real-audio
 - Web-Cache, Firewall, Security, DDoS-Schutz, Intrusion Detection, etc.
- **Erweiterung des i3-Prinzips**
 - ▶ Paket wird an Folge (stack) von IDs gesendet (wie IP-Source-Routing)
 - ▶ Ziel eines Triggers = Folge von IDs bzw. Peer-Adresse



Verkettung von Diensten mit i3 – II

- **Beispiel für Service Composition**

- ▶ Sender sendet MPEG-Layer 2 (192 kbps)
- ▶ R_1 will MP2 mit geringer Bandbreite, R_2 will Real-Audio
- ▶ S_1 bzw. S_2 setzen Formate um – Sender bemerkt davon nichts



- **Wie findet man einen bestimmten Dienst oder Inhalt?**

- ▶ Übliche Vorgehensweise:

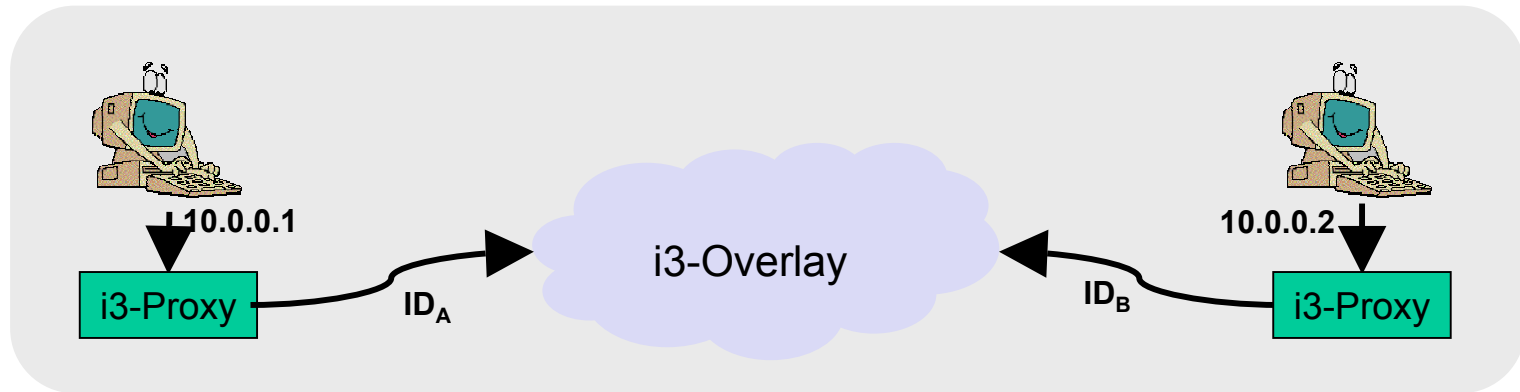
Abbildung des Inhalts/Dienstes per Hash-Funktion

- $H(\text{„WWW: www.spiegel.de“}) \rightarrow ID_1$
 - $H(\text{„SERVICE: HTML to WML“}) \rightarrow ID_2$
 - $H(\text{„SERVICE: REALAUDIO to MP2“}) \rightarrow ID_3$
 - $H(\text{„SERVICE: INTRUSION DETECTION“}) \rightarrow ID_4$
 - $H(\text{„RADIO: SWR-3 / TYPE: MP2-192 “}) \rightarrow ID_5$
 - $H(\text{„NAME: Erwin Pelzig“}) \rightarrow ID_6$

- **i3 bietet interessante Eigenschaften**
 - ▶ Skalierbar
 - ▶ Flexibel (beliebig) kombinierbar
- **Probleme**
 - ▶ Neuer Adressraum
 - ▶ Keine Standardschnittstellen
 - ▶ Standardanwendungen müssen modifiziert werden
(gleiches Problem wie Migration zu IPv6)
- **Lösungsansatz:**
 - ▶ i3-Proxy verbindet IP-Anwendungen mit i3-Overlay

i3-Proxy zur Unterstützung von Standardanwendungen

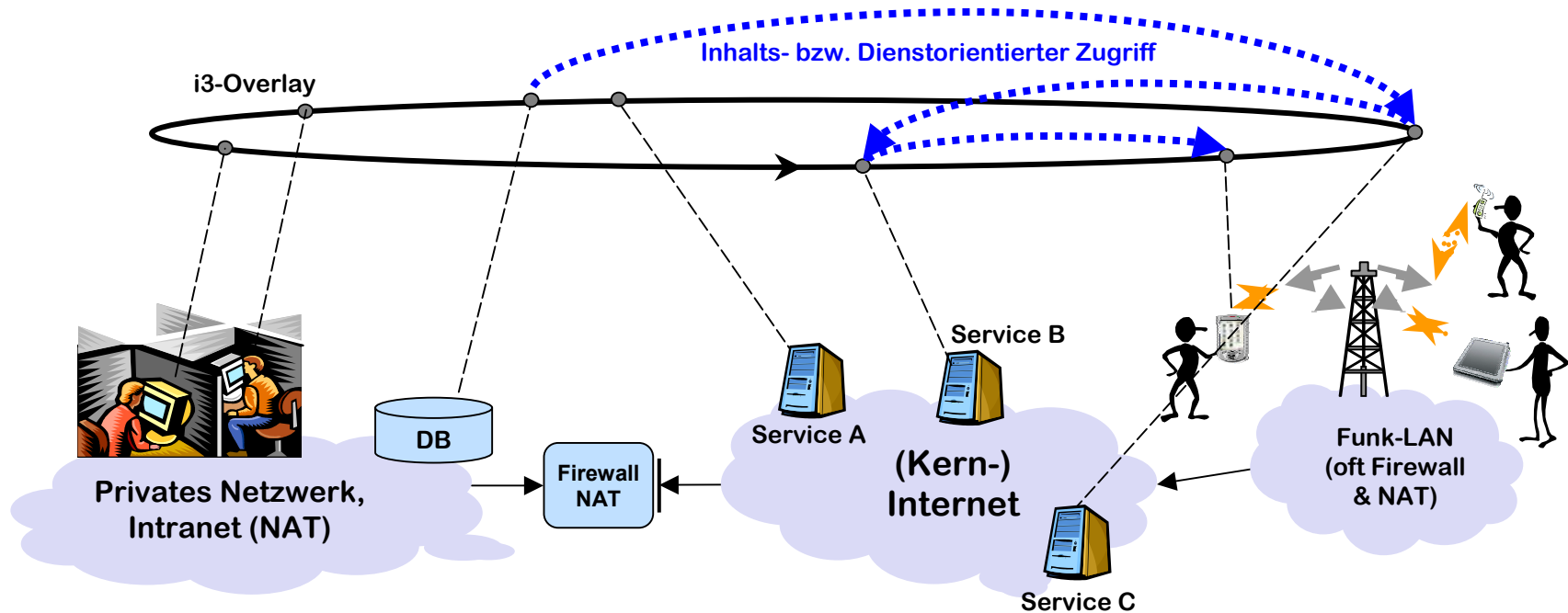
- **Umsetzung der Adressierung:**
 - ▶ Abbildung fiktiver DNS-Namen auf i3-IDs
 - ▶ „Network Address Translation“ (NAT) für Overlay-Netzwerke



- **Ablauf:**
 1. i3-Proxy analysiert DNS-Anfragen
 2. Anfrage mit „i3“ wird auf i3-ID abgebildet :
 - z.B.: `www.cnn.i3` → `Hash(„www.cnn.i3“) = IDA`
 3. Aufbau einer privaten i3-Verbindung mit gesuchtem Partner
 4. Rückgabe einer fiktiven IP-Adresse an Standardanwendung
 5. Komplette Kommunikation erfolgt nun über i3
 - Mit allen Features der i3-Architektur

Inhaltsorientierte Überbrückung der Heterogenität

- **Verbindung heterogener Netzwerke, Geräte und Anwendungen**
 - ▶ **Mobilität und Sicherheit**
 - transparent für Benutzer und diejenigen, die ihn erreichen wollen
 - ▶ **Automatische Komposition der entsprechenden Dienste**
 - z.B. Wechsel von Formaten



- **Internet Technologie**

- ▶ Grenzen werden zunehmend sichtbar und kritisch
 - Mangelnde Unterstützung für Kommunikationsformen
 - Probleme durch NAT, Firewalls, ...

- **Ansatz der Indirektion zeigt neue Möglichkeiten auf**

- ▶ Kommunikationsformen
- ▶ Flexible Komposition von Diensten
- ▶ Einbindung von Standardanwendungen

- **Umsetzung**

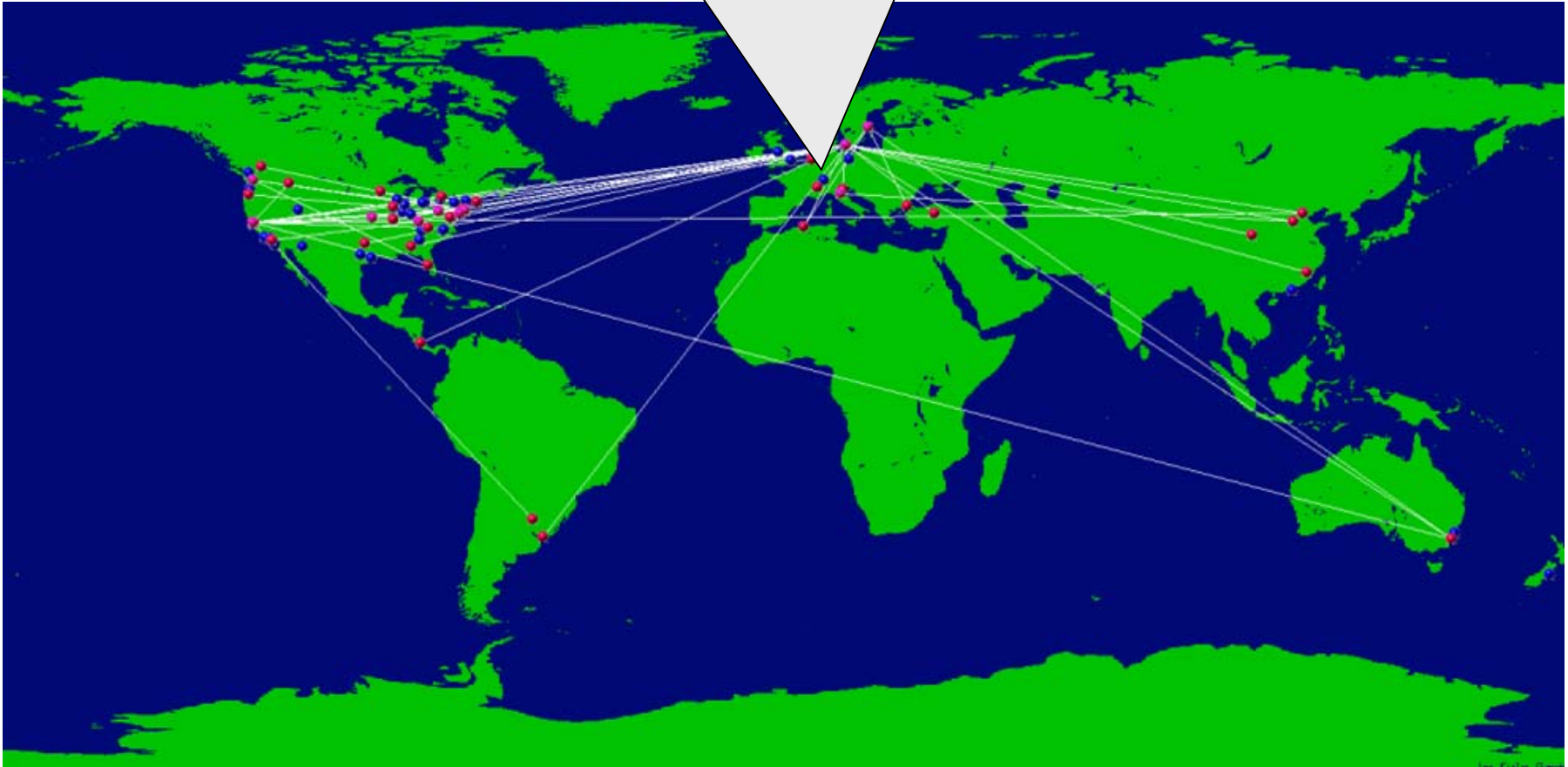
- ▶ Einbindung mobiler Firmenmitarbeiter
- ▶ Projekt mit der Landesstiftung Baden-Württemberg

i3-Testbed in Planet Lab

`Peace.ri.uni-tuebingen.de`
`Freedom.ri.uni-tuebingen.de`

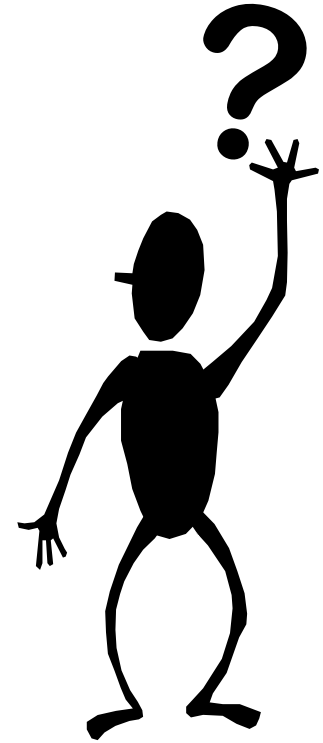


PLANETLAB



Vielen Dank !

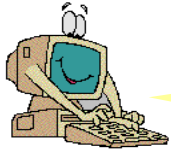
Fragen ?



<http://net.informatik.uni-tuebingen.de/PS/>

- **Indirektionsmechanismus unterstützt zahlreiche Anwendungsfälle**

- ▶ Erster Ansatz für generische P2P-Dienste
- ▶ Anwendungsgebiete
 - Multicast, Anycast, Mobilität, Komposition von Diensten
 - Weitere Anwendungen (geplant / teilweise realisiert):
Security, DNS-Ersatz, VPNs, verteilte Firewall, ...
- ▶ Unterstützung für “Legacy Applications”
- ▶ Keine schlechte Performance trotz Indirektion und DHT



DNS request:
IP address of
thinkpad.klaus.i3 ?

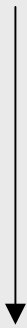
Host A: 134.2.11.12

thinkpad.klaus.i3



Host B: 128.2.14.20

1. DNS Request (*thinkpad.klaus.i3*)



Proxy A

Proxy B

$H(\textit{thinkpad.klaus.i3})$

i3 overlay



Host A: 134.2.11.12

DNS request:
IP address of
thinkpad.klaus.i3 ?

thinkpad.klaus.i3



Host B: 128.2.14.20

1. DNS Request (*thinkpad.klaus.i3*)

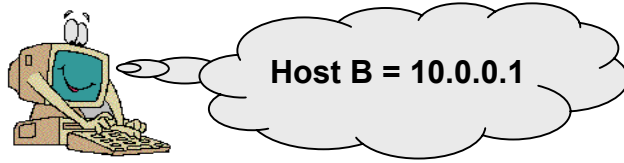
Proxy A

Hash(*thinkpad.klaus.i3*):
Initiate private
i3 connection

Proxy B

H(*thinkpad.klaus.i3*)

i3 overlay



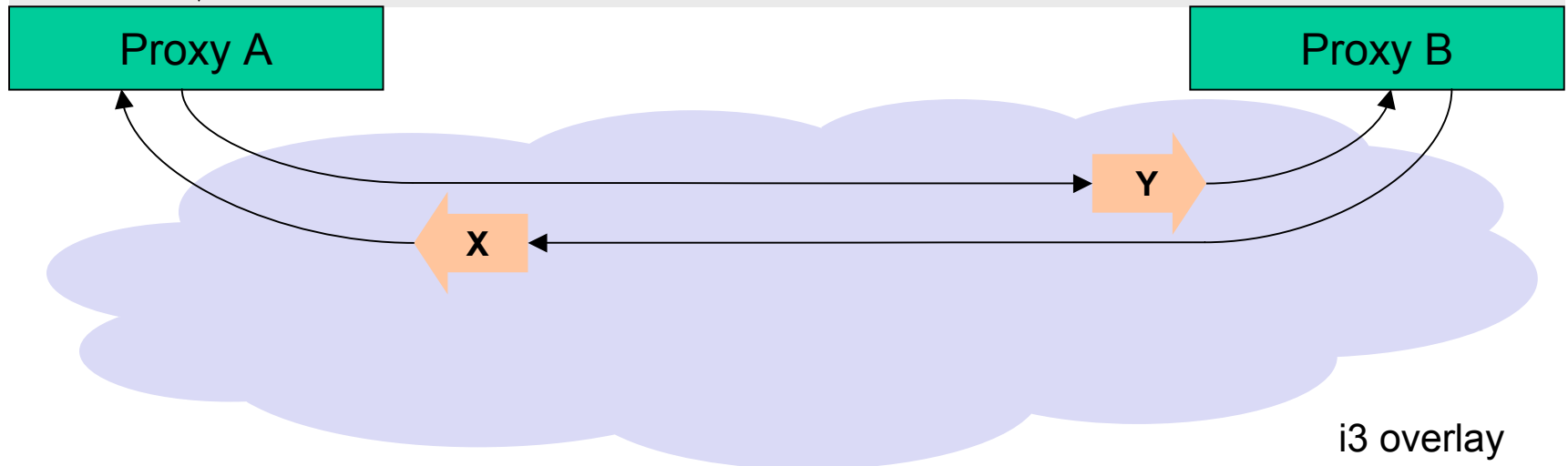
Host A: 134.2.11.12

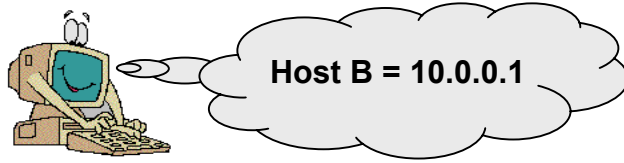
thinkpad.klaus.i3



Host B: 128.2.14.20

1. DNS Request (*thinkpad.klaus.i3*)
2. DNS Response: **10.0.0.1**





Host A: 134.2.11.12

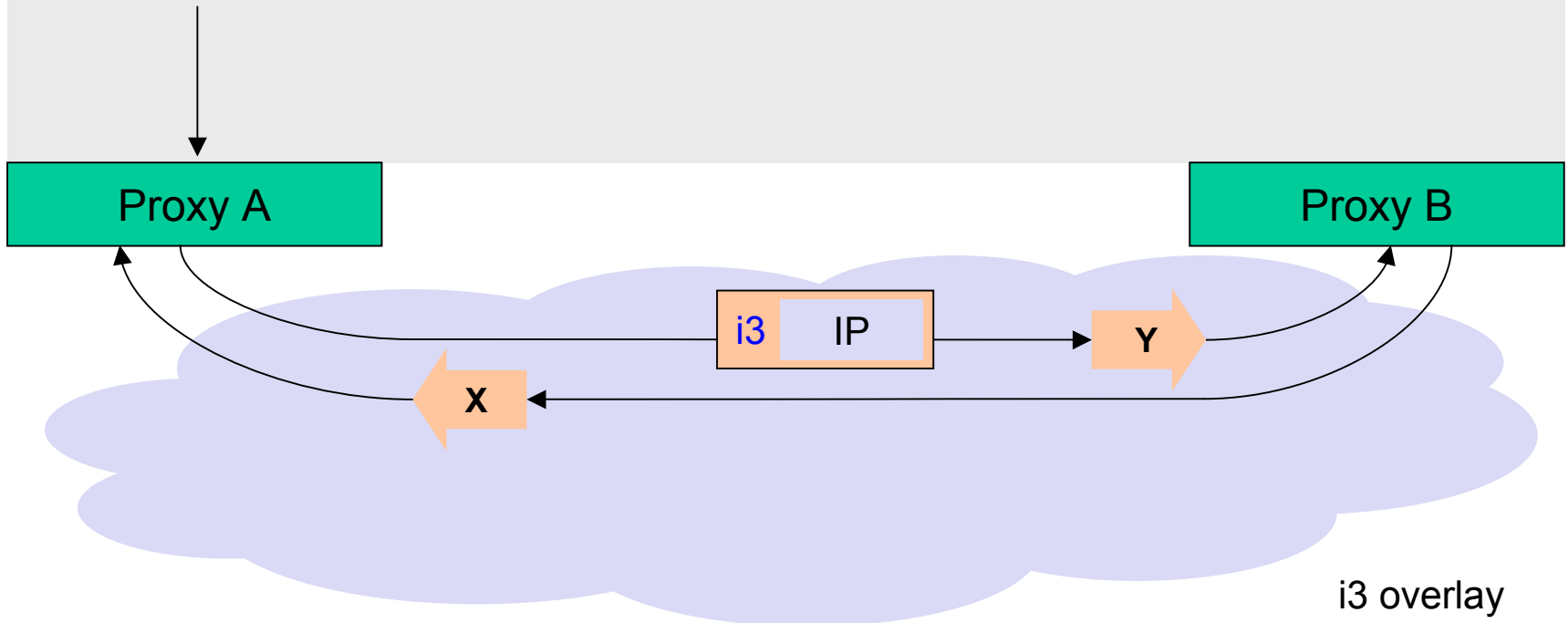
thinkpad.klaus.i3

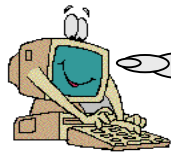


Host B: 128.2.14.20

1. DNS Request (*thinkpad.klaus.i3*)
2. DNS Response: 10.0.0.1

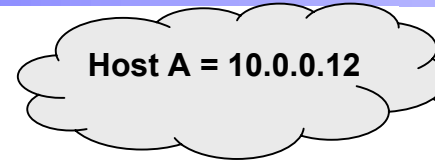
n: IP: 134.2.11.12 → **10.0.0.1**





Host B = 10.0.0.1

Host A: 134.2.11.12



Host A = 10.0.0.12

thinkpad.klaus.i3

Host B: 128.2.14.20

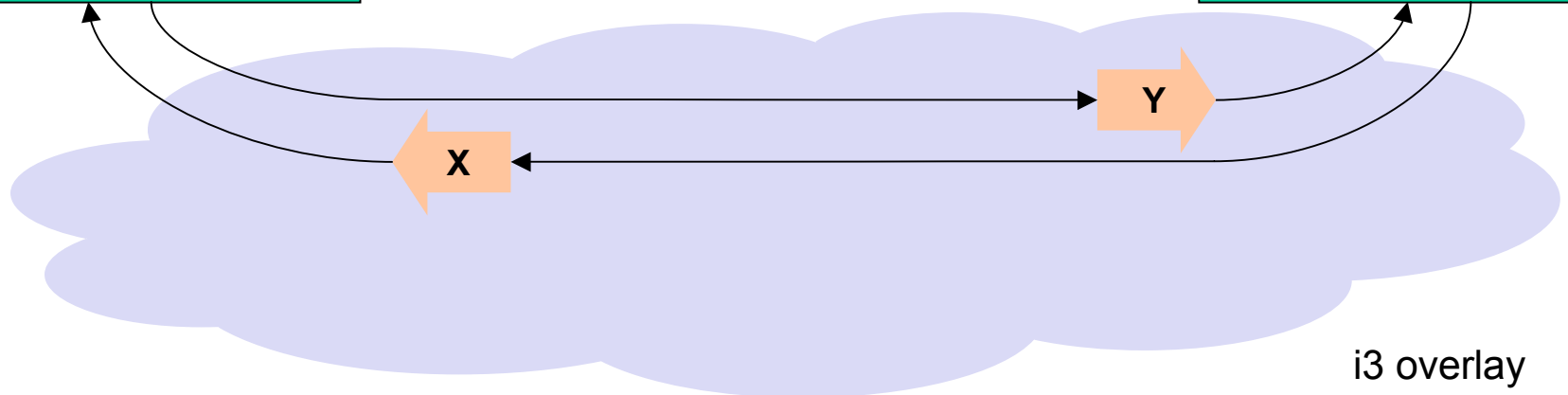
1. DNS Request (*thinkpad.klaus.i3*)
2. DNS Response: 10.0.0.1

n: IP: 134.2.11.12 → 10.0.0.1

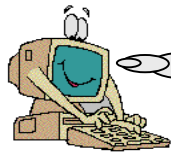
n+1: IP: **10.0.0.12** → 128.2.14.20

Proxy A

Proxy B



i3 overlay



Host B = 10.0.0.1

Host A: 134.2.11.12



Host A = 10.0.0.12

thinkpad.klaus.i3

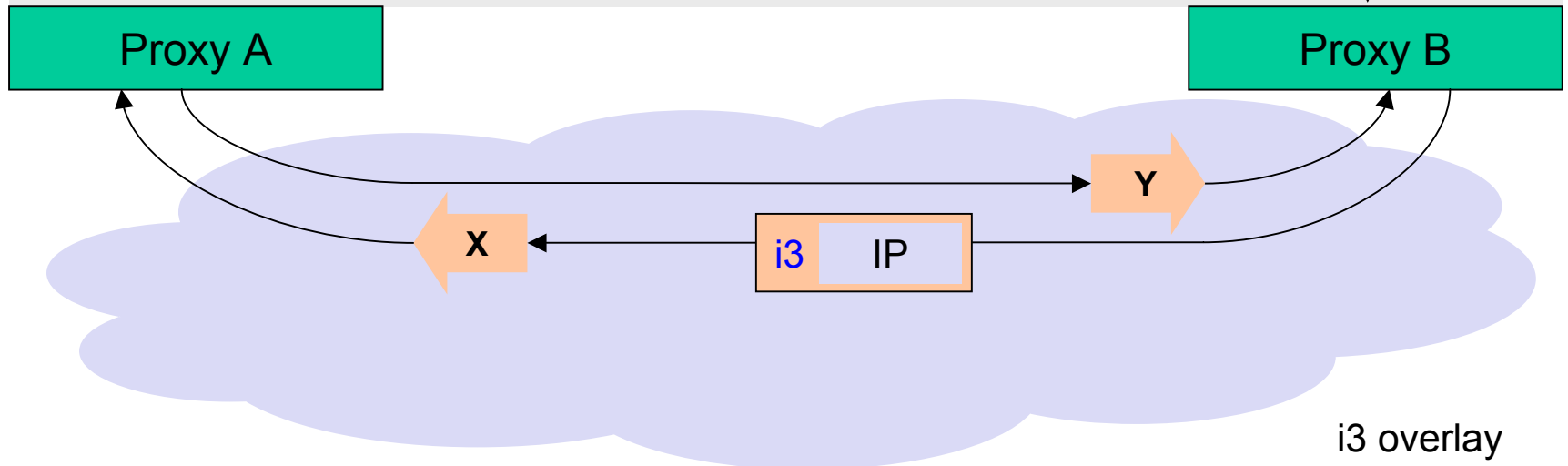
Host B: 128.2.14.20

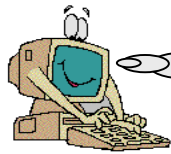
1. DNS Request (*thinkpad.klaus.i3*)
2. DNS Response: 10.0.0.1

n: IP: 134.2.11.12 → 10.0.0.1

n+1: IP: 10.0.0.12 → 128.2.14.20

m: IP: **10.0.0.12** ← 128.2.14.20





Host B = 10.0.0.1

Host A: 134.2.11.12



Host A = 10.0.0.12

Host B: 128.2.14.20

thinkpad.klaus.i3

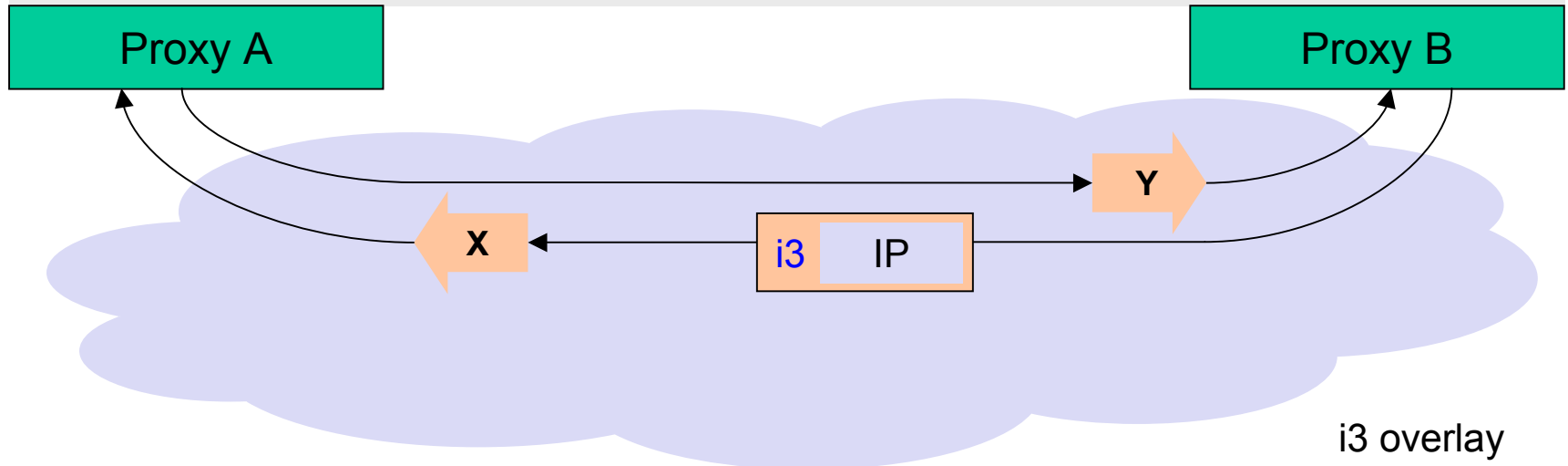
1. DNS Request (*thinkpad.klaus.i3*)
2. DNS Response: 10.0.0.1

n: IP: 134.2.11.12 \rightarrow 10.0.0.1

n+1: IP: 10.0.0.12 \rightarrow 128.2.14.20

m: IP: 10.0.0.12 \leftarrow 128.2.14.20

m+1: IP: 134.2.11.12 \leftarrow **10.0.0.1**



Ablauf (detailliert)

1. **DNS-Anfrage: www.cnn.i3**
 - ▶ Wird im Proxy abgefangen und auf $H(\text{www.cnn.i3})$ abgebildet
 2. **Proxy fragt über $H(\text{www.cnn.i3})$ beim Server bzgl. privater i3-Verbindung an**
 - ▶ Dabei wird ein privater Trigger ID_C für den Client (Anfragenden) eingerichtet
 - ▶ Server richtet privaten Trigger ID_S ein und antwortet über ID_C
 3. **Proxy speichert Infos zur privaten Verbindung und antwortet der Anwendung mit einer fiktiven IP-Adresse (10.a.b.c)**
 - ▶ Abbildung 10.a.b.c auf ID_S wird im Proxy gespeichert
 4. **Alle Pakete der Anwendung an 10.a.b.c werden an ID_S gesendet**
 - ▶ Alle eintreffenden Pakete mit Ziel ID_C werden als 10.a.b.c-Pakete an Anwendung übergeben
-
- **Anwendung bemerkt die Verwendung des Overlay-Netzes nicht**
 - **DNS wird für Umsetzung der Adressformate verwendet**
 - ▶ Eigene DNS-Namensräume werden möglich

Verkettung von Diensten mit i3 – II

- **i3-Zieladresse (in Trigger):**

- ▶ Adressierung eines Rechners:

IP / Port

- ▶ Adressierung einer ID:

ID

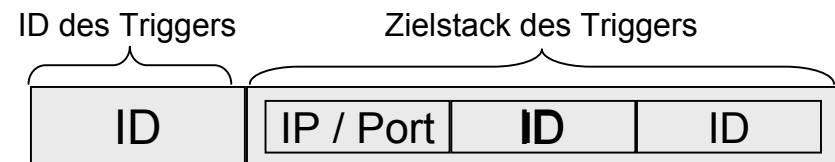
- **i3-Stack:**

- ▶ Stapel mit i3-Adressen:

IP / Port	ID	ID
-----------	----	----

- **I3-Trigger:**

- ▶ ID und i3-Stack:



- **Wegewahl:**

- ▶ Wird ein Ziel (Rechner) erreicht, wird das entsprechende Adressfeld aus dem Stack gestrichen.
 - ▶ Wird ein Trigger erreicht, wird die entsprechende ID im Stack durch den Ziel-Stack des Triggers ersetzt.

Ablauf (detailliert)

1. **DNS-Anfrage: www.cnn.i3**
 - ▶ Wird im Proxy abgefangen und auf $H(\text{www.cnn.i3})$ abgebildet
 2. **Proxy fragt über $H(\text{www.cnn.i3})$ beim Server bzgl. privater i3-Verbindung an**
 - ▶ Dabei wird ein privater Trigger ID_C für den Client (Anfragenden) eingerichtet
 - ▶ Server richtet privaten Trigger ID_S ein und antwortet über ID_C
 3. **Proxy speichert Infos zur privaten Verbindung und antwortet der Anwendung mit einer fiktiven IP-Adresse (10.a.b.c)**
 - ▶ Abbildung 10.a.b.c auf ID_S wird im Proxy gespeichert
 4. **Alle Pakete der Anwendung an 10.a.b.c werden an ID_S gesendet**
 - ▶ Alle eintreffenden Pakete mit Ziel ID_C werden als 10.a.b.c-Pakete an Anwendung übergeben
-
- **Anwendung bemerkt die Verwendung des Overlay-Netzes nicht**
 - **DNS wird für Umsetzung der Adressformate verwendet**
 - ▶ Eigene DNS-Namensräume werden möglich