JXTA (und Poblano): Eine Protokoll-architektur zur Realisierung flexibler P2P-Applikationen

Marco Winter
simawint@stud.informatik.uni-erlangen.de

Überblick

- JXTA: Definition, Besonderheiten, Entwicklung
  - Konzepte und Protokolle
  - Aufbauende Softwarearchitektur
- Poblano
  - Vertrauensverhältnisse, Vergleich zu anderen Modellen
  - Beispiel: Schlüsselsuche
  - Zertifikate

Einleitung

- Was ist JXTA?
  - Protokollarchitektur für P2P-Anwendungen
  - Definition von Konzepten und Protokollen
  - Aufbau eines virtuellen P2P-Netzwerkes

Einleitung (Forts.)

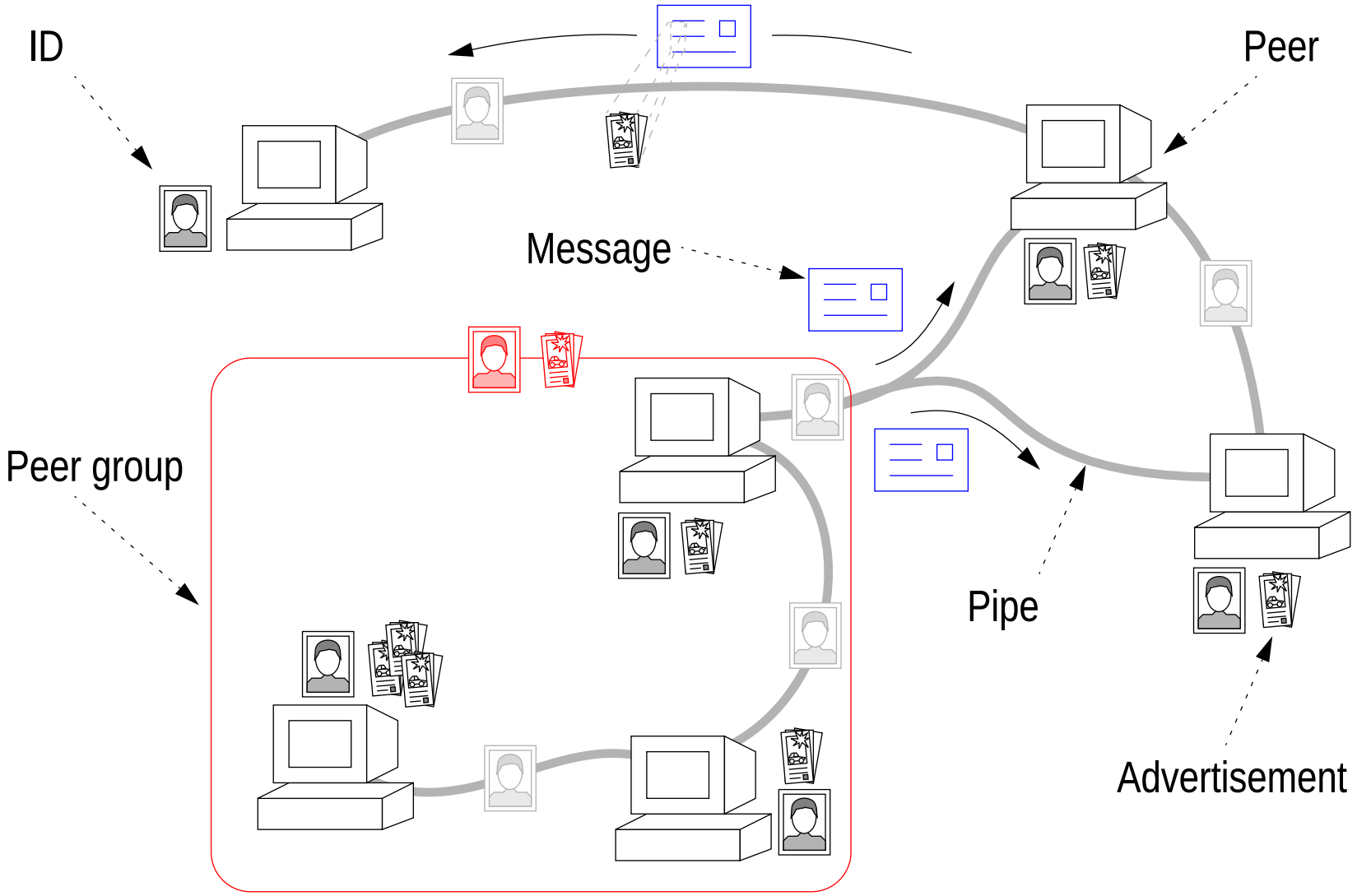
- Besonderheiten von JXTA
  - Unabhängigkeit von speziellen Betriebssystemen, Programmiersprachen und Netzwerktopologien
  - Theoretisch einsetzbar auf PCs, Handys, PDAs, ...
  - Ein JXTA-Knoten kann alles Mögliche sein!

Einleitung (Forts.)

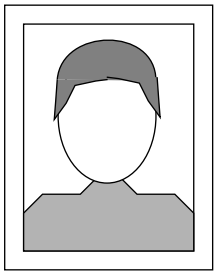
Project JXTA

- Entwicklung von JXTA
  - Von Sun Microsystems Inc. ins Leben gerufen
  - Mitte April 2001 als Open Source Projekt ins Netz gestellt, inkl. Java-Implementierung von Sun
  - Wird von der JXTA Community ständig weiterentwickelt; Mittlerweile Implementierungen für C, Perl, Python, ...
  - Projekt Homepage unter `www.jxta.org` (mit Sourcecode, stable Releases, Demos, ...)

Konzepte in JXTA: Überblick



ID

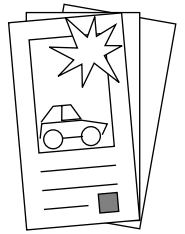


- Eindeutiger Bezeichner im weltweiten Netz
- Nur eine eindeutige ID für jede Ressource
- Verwendete Implementierung:
  - URN Representation
  - 128-320 bit UUID

`urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206`

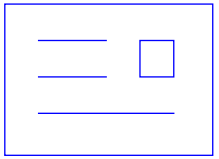
Advertisement

- “Visitenkarte” bzw. Eigenwerbung für Ressourcen im Netz
- begrenzte Lebensdauer
- in XML codiert: flexibel und erweiterbar



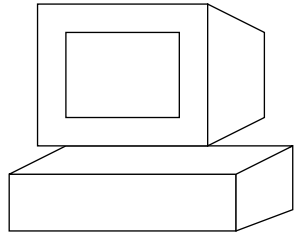
```
<?xml version=”1.0”?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta=”http://jxta.org”>
    <GID> urn:jxta:jxta-NetGroup</GID>
    <MSID>
    urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206
    </MSID>
    <Name>NetPeerGroup</Name>
    <Desc>NetPeerGroup by default</Desc>
</jxta:PGA>
```

Message

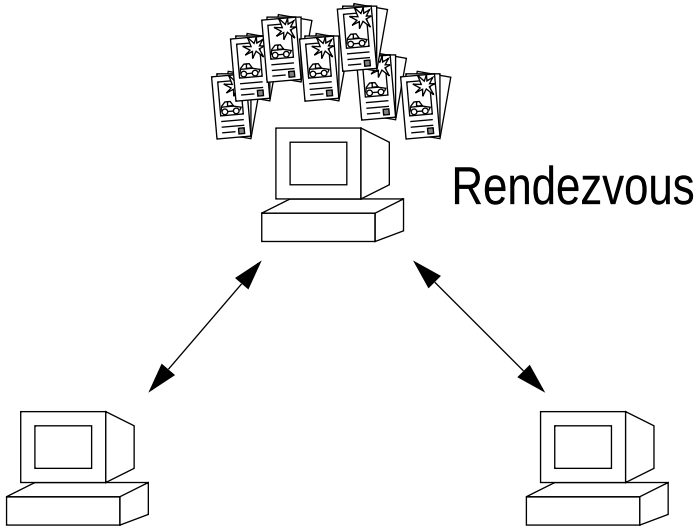


- universelles Nachrichtenformat für binäre und Textinhalte
- speichert auch Metainformationen
  - Routing
  - Identifizierung des Senders
  - Entschlüsselung der Nachricht
  - ...

Peer

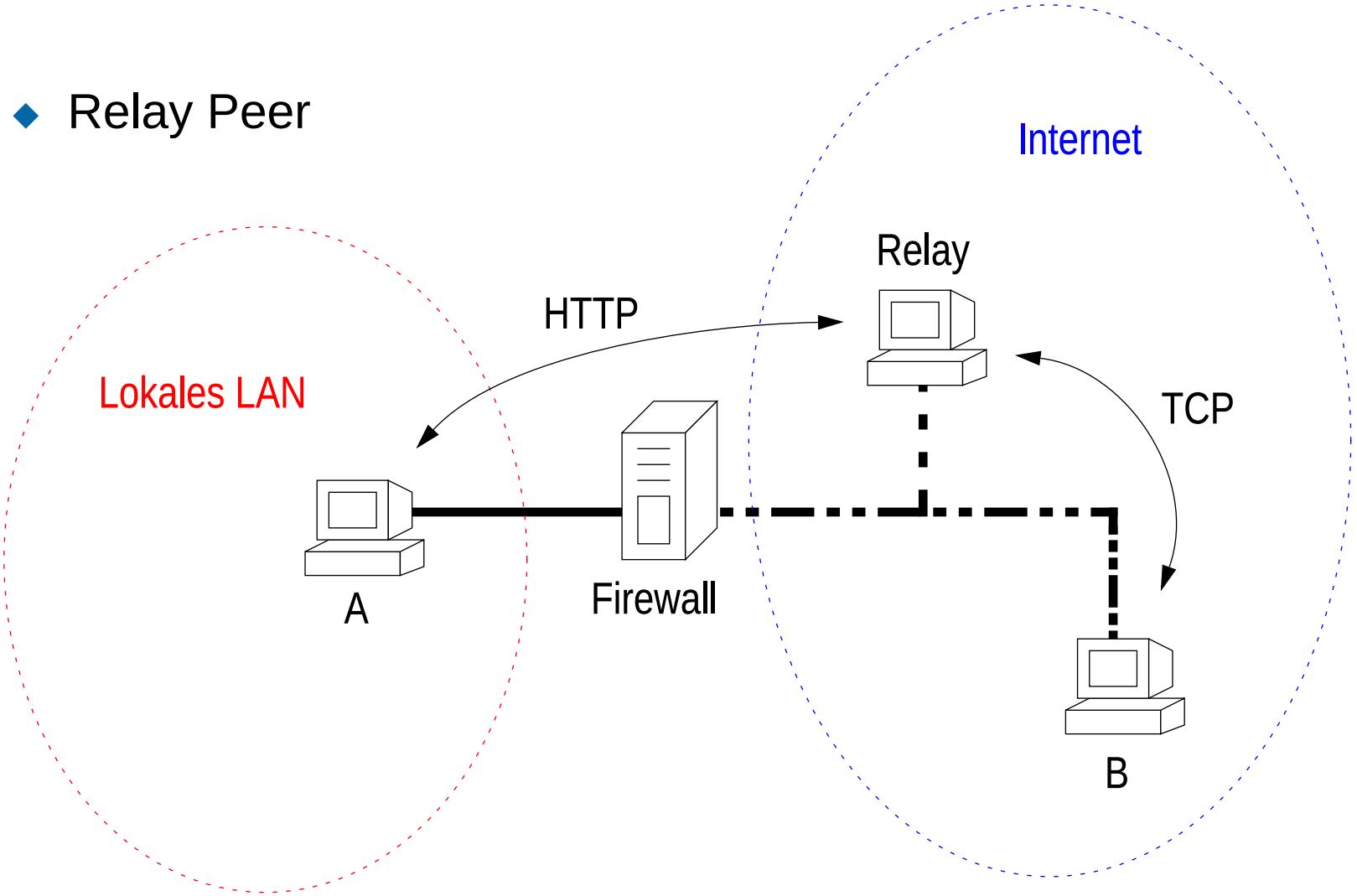


- Name für Knoten im System
- Knoten kann Rechner, Prozess, oder andere Einheit sein
- Spezielle Peers:
  - Rendezvous Peer

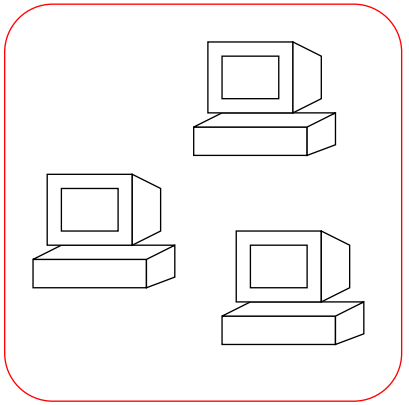


Peer (Fortsetzung)

- Relay Peer

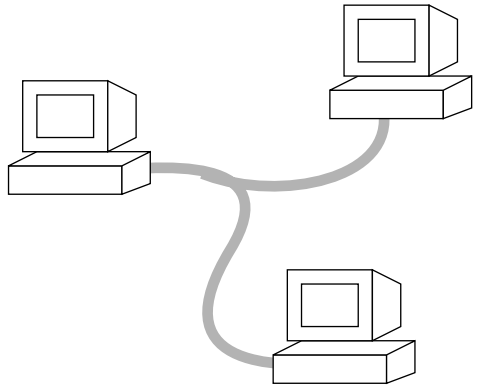


Peer Group



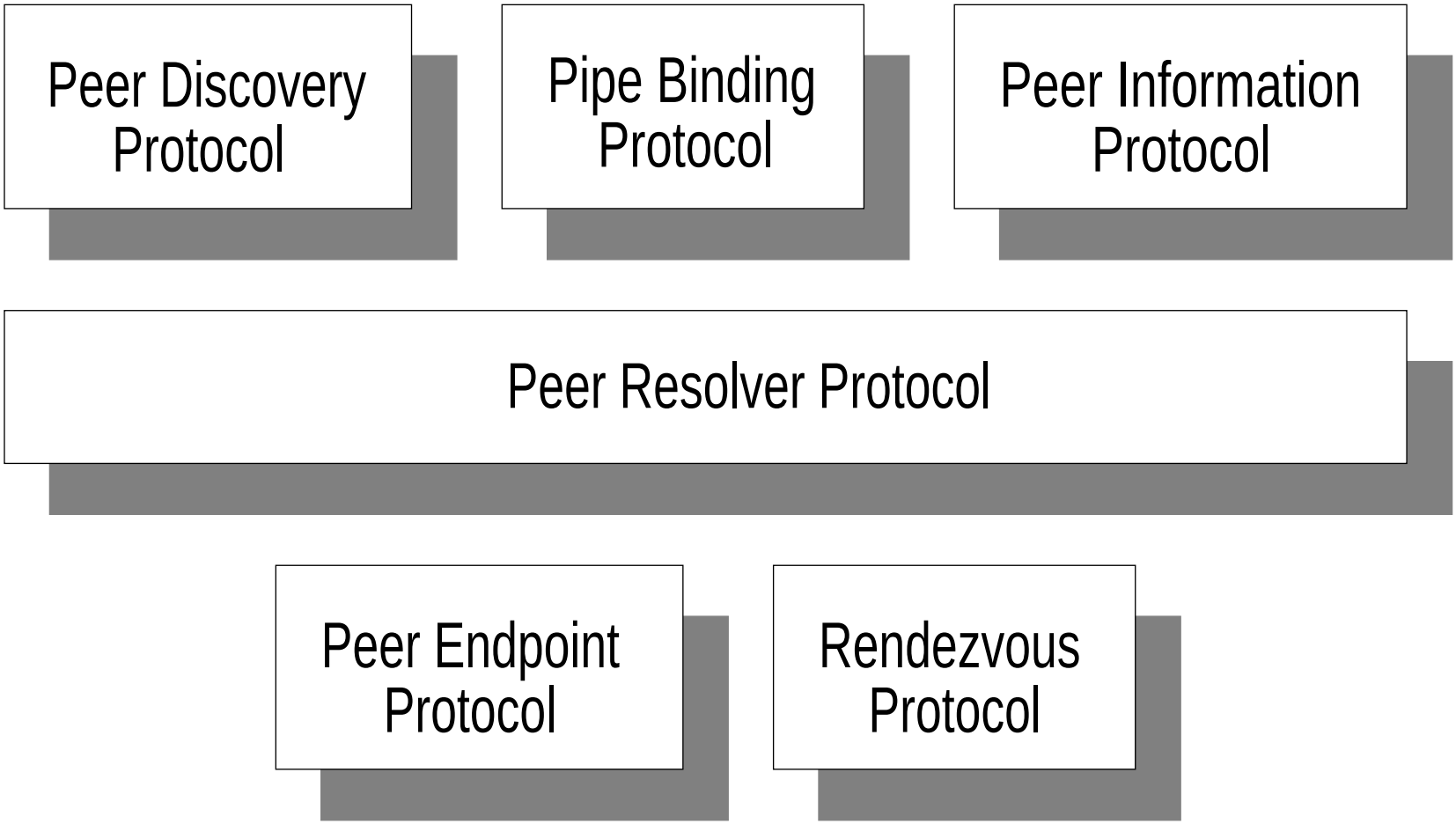
- Mehrere Peers können zu einer Peer Group zusammengefasst werden
- Jede Gruppe schreibt gemeinsame Standards vor
- Ein Peer kann Mitglied beliebig vieler Gruppen sein
- Jeder Peer muss die Protokolle “seiner” Gruppen implementieren
- Jeder Peer gehört der *NetPeerGroup* an

Pipe



- Virtueller, unidirektionaler Kanal zwischen zwei oder mehreren Peers
- Ein- und Ausgänge können dynamisch an verschiedene Endpunkte gebunden werden
- Zwei Versionen von Pipes:
  - point-to-point Pipe
  - Propagation Pipe

Protokolle in JXTA:Überblick



Protokolle in JXTA

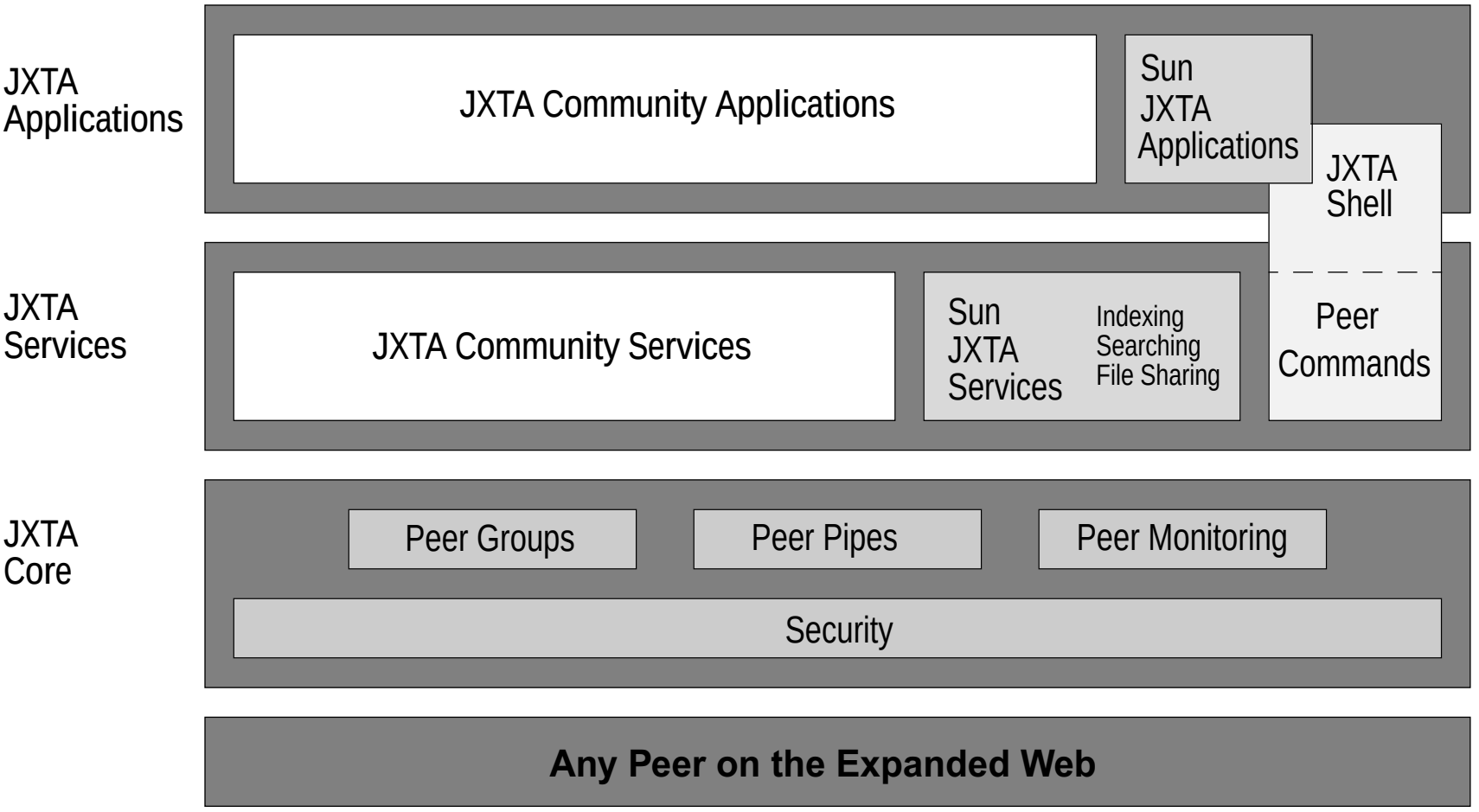
- Allgemeines
  - Grundprotokolle, werden von der *NetPeerGroup* gefordert
  - arbeiten auf unidirektionalen, unzuverlässigen Verbindungen
  - Sollten von jedem Peer implementiert werden
  - Eigene Protokolle können hinzugefügt werden

- Die Protokolle im Einzelnen
  - **Peer Endpoint Protokoll**
    - findet einen Weg von A nach B, evtl. mit Hilfe von Relays
  - **Rendezvous Protokoll**
    - Nötig, um Rendezvous Peer zu realisieren
    - Implementiert einen Nachrichtenverteiler

- **Peer Resolver Protokoll**
  - Steuert Durchführung von Anfragen und Antworten
  - Erlaubt Einbindung von “Handlern” für bestimmte Anfragetypen
- **Peer Discovery Protokoll**
  - Zur Verbreitung von eigenen und zum Finden von fremden Advertisements

- **Peer Information Protokoll**
  - Erlaubt die Abfrage von Statusinformationen von anderen Knoten, z.B. Auslastung, Fehlerrate, ...
- **Pipe Binding Protokoll**
  - Aufbau einer Pipe zwischen zwei oder mehreren Knoten
  - Verbindet die Enden der Pipe mit den gewünschten Endpunkten

Architektur von JXTA: Überblick



- Die Schichten im Einzelnen
  - JXTA Core
    - Eigentliche Implementation von JXTA
    - Nur Mechanismen, keine Policies!
  - JXTA Services
    - Komplexere Dienste, aufbauend auf JXTA

- JXTA Applications
  - Kann Dienste der unteren Schichten in Anspruch nehmen
  - Keine strikte Kapselung der Schichten

JXTA Shell

- zeilenorientierter Kommandointerpreter
- bietet Einblicke in den Kern von JXTA
- grundlegende Abläufe können manuell erledigt werden
- Von Unix bekannte Konzepte:
  - Unix-Pipes
  - Skriptverarbeitung
- Wird automatisch mit JXTA “gebootet”

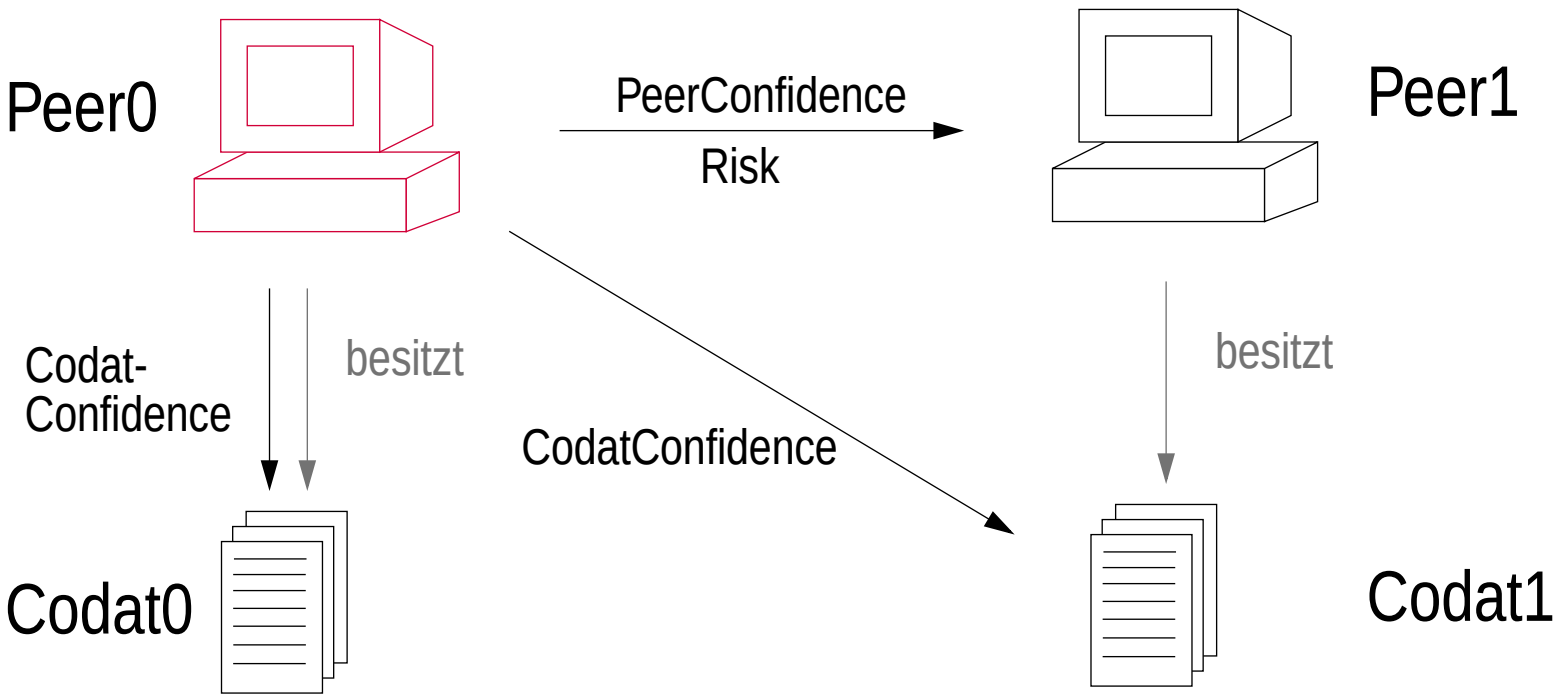
Poblano: Einleitung

- Vertrauensmodell für JXTA
- Abbildung der Realität: Personen vertrauen einander
- Jeder Peer besitzt Meinungen über andere Peers
- Jeder Peer erwirbt einen gewissen Ruf
  - Vertrauens- /Misstrauensbasis

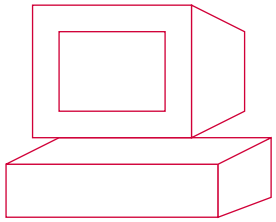
Vergleich mit anderen Modellen

- Andere:
  - Vertrauensbasis: Leistung, Zuverlässigkeit, Ehrlichkeit,...
  - Kein Vertrauensschema für angebotene Daten und Dienste
- In Poblano:
  - Neue Komponente: “Zufriedenheit” mit Daten (Codats)
  - 3 Komponenten:
    - **CodatConfidence**: Zufriedenheit des Benutzers mit Codat
    - **PeerConfidence**: Zufriedenheit des Benutzers mit Peer
    - **Risk**: Traditionelle Vertrauenswerte

Vertrauensverhältnisse



Interner Aufbau



Risk-Tabelle			
Peer 1	Peer 2	...	Peer u

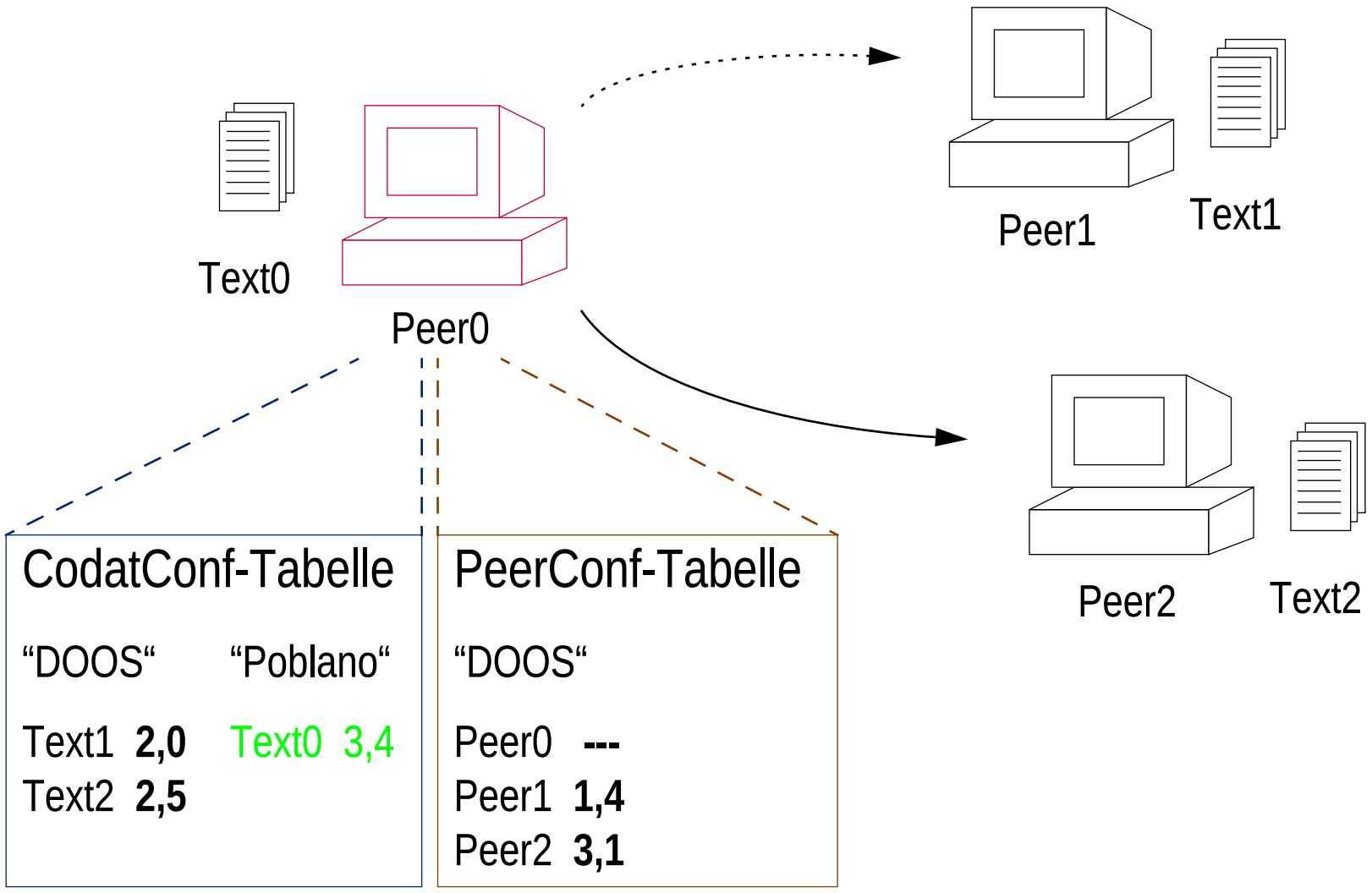
CodatConfidence-Tabelle 1	PeerConfidence-Tabelle 1
CodatConfidence-Tabelle 2	PeerConfidence-Tabelle 2
...	...
CodatConfidence-Tabelle i	PeerConfidence-Tabelle i

Key 1	Key 2		Key n
Codat1,1	Codat2,1		Codatn,1
Codat1,2	Codat2,2	...	Codatn,2
...	...		...
Codat1,j	Codat2,k		Codatn,m

Key 1	Key 2		Key n
Peer1,1	Peer2,1		Peern,1
Peer1,2	Peer2,2	...	Peern,2
...	...		...
Peer1,r	Peer2,s		Peern,t

Beispiel: Schlüsselsuche

- Interner Aufbau:



Beispiel (Forts.)

- Ablauf:

(1) Peer0 sucht in eigener CodatConfidence-Tabelle nach lokalem gültigen Eintrag

(2) Falls nicht gefunden, werden zuverlässige Peers aus der PeerConfidence-Tabelle angesprochen und gefragt

(3) Schritte 1 und 2 werden wiederholt, bis ein Peer passende lokale Daten findet. Die Daten werden übertragen, und die internen Tabellen aktualisiert

Beispiel (Forts.)

- Berechnungen
  - Jede Komponente wird durch Werte aus einem bestimmten Zahlenbereich bewertet

Wert	Bedeutung
-1	Misstrauen
0	Ignorieren
1	geringes Vertrauen
2	durchschn. Vertrauen
3	gutes Vertrauen
4	volles Vertrauen

Beispiel (Forts.)

- Neuberechnung der Zufriedenheit mit dem Provider

$$newPeerConf = \left(oldPeerConf + \frac{1}{|K|} \sum_{a \in K} CodatConf_{Provider} \right) / 2$$

K = Anzahl an CodatConfidences des Providers

- Neubewertung der Zufriedenheit mit eigenen Codats

$$newCodatConf = \left(oldCodatConf + feedback \times \frac{PeerConf_{Initiator}}{4} \right) / 2$$

Kooperations-Schwellwert

- Eigentliches Entscheidungskriterium für die Vertrauens-würdigkeit eines Peers
- Benutzerdefinierter Wert: “Wichtigkeit”

$$\mathrm{PeerConf} \times \mathrm{Importance} > \frac{\mathrm{Risk}_{\mathrm{Peer}}}{\frac{1}{|K|}\sum_{a \in K} \mathrm{CodatConf}_{\mathrm{Peer}}}$$

Zertifikate in Poblano

- Anwendungsgebiete in JXTA
  - Privilegierter Zugang zu Peergruppen, Codats, ...
- Poblano: Unterstützung eines breiten Spektrums von Zertifikaten
  - Selbstsigniert
  - beglaubigt bzw. fremdsigniert
  - von Certificate Authorithy (CA) signiert

Zertifikate in Poblano (Forts.)

- Konkrete Implementierung
  - Spezieller Fall der Suche nach Schlüsselworten
  - Eigene **CertConfidence**-Tabelle, ähnlich CodatConfidence
  - Neue **PeerConfidence**-Tabelle
    - Bewertung für Peer als Besitzer
    - Bewertung für Peer als Fürsprecher

Zusammenfassung

- JXTA
  - Systemarchitektur zur Entwicklung von P2P-Anwendungen
  - Definition von Konzepten und Protokollen
- Poblano
  - Vertrauensmodell für JXTA
  - Bewertung von Peers aufgrund von Leistung und Zufriedenheit