

# Netze und Protokolle für das Internet



## 9. Aktive Netze

### Inhalt

- Passive/Active Networking
- Modell und Typen aktiver Netze
- Einführungsarchitekturen
- Architektur eines aktiven Knotens
  - Aktive Anwendung (AA)
  - Ausführungsumgebung
  - NodeOS
  - Komponenten im Datenpfad
- De/Multiplexing
  - Active Network Encapsulation Protocol
- Beispiele aktiver Anwendungen
  - Smart Packet Dropping
  - Multicast
  - Web-Caching
  - Manipulation von Datenströmen
  - Intelligentes Routing
  - Aktives Management
- Sicherheit
- Ende-zu-Ende Argument

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

2

### Motivation

- Probleme
  - Einführung neuer Protokolle und Netz-Dienste ist langwierig und nicht einfach, z.B. IP Multicast, IPv6.
  - Konfiguration und Management von Netzelementen ist meist wenig flexibel.
- Ziel: Architektur zum
  - schnellen,
  - globalen (nicht auf jedem Knoten) und
  - dynamischen
 Einführen bzw. Testen innovativer, individueller Dienste  
 ⇒ programmierbare Netzelemente

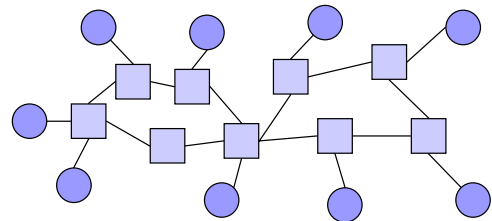
SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

3

### Passive Networking

- intelligente Endsysteme am Rand des Netzes
- passive Netzelemente (Switches, Router) im Innern des Netzes
- Beispiele: PSTN, Internet



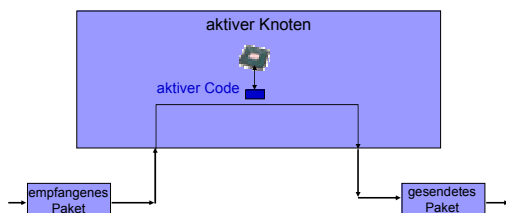
SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

4

### Active Networking

- Verarbeitung
  - store
  - compute
  - forward



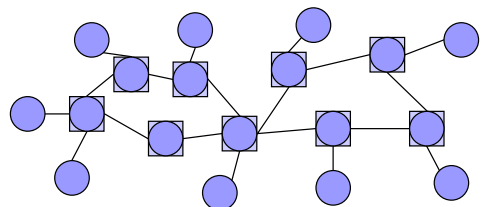
SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

5

### Modell Aktiver Netze

- Pakete können das Verhalten der Netzelemente "on-the-fly" ändern.
  - aktive Pakete (in-band)
  - aktive Erweiterungen (out-of-band)



SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

6

## Active Networking Typen

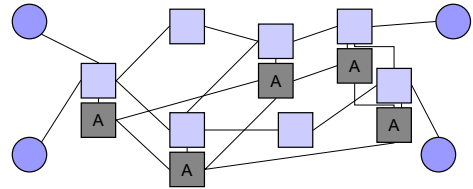
- moderat
  - Code wird durch Service Provider zur Verfügung gestellt.
    - Beispiel: Intelligente Netze
  - Benutzer selektieren verfügbaren Code.
  - erlaubt die schnelle Einführung neuer Protokolle und Dienste
- intensiv
  - Benutzer injiziert eigenen Code in aktiven (Daten-)Paketen (Capsules).
  - Knoten führt Capsules in Execution Environment (EE) aus.

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

7

## Einführungsarchitektur: Overlay



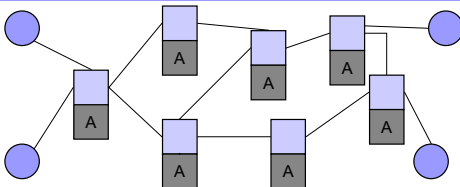
- separates Overlay-Netz mit Intelligenz
- Dienste müssen out-of-band eingerichtet werden.
- notwendig, wenn Knoten nicht erweitert werden können.
- vgl. Intelligentes Netz
- teuer und inflexibel

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

8

## Eingebettete Intelligenz



- eingebettete Intelligenz in ausgewählten (oder allen) Netzelementen
- Einrichtung der Dienste kann in-band erfolgen.
- erlaubt inkrementelles und selektives Einführen von Diensten

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

9

## Granularität aktiver Verarbeitung

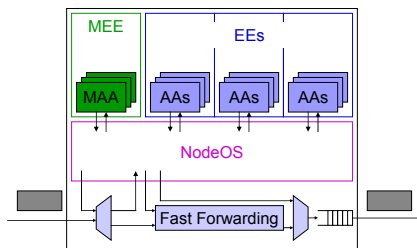
- Control Plane
  - global
    - betrifft den Knoten als Ganzes
    - z.B. Änderungen beim Routing
  - pro Fluss (flow)
    - erfordert nicht-transiente Speichermöglichkeiten (Rendezvous v. Capsules)
    - z.B. MPLS- o. RSVP-Flüsse
  - pro Paket
    - z.B. aktive Staukontrolle
- Data Plane
  - Änderung der Paketgröße und der Nutzlast
    - z.B. Transcoding
  - sehr Ressourcen-intensiv

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

10

## Architektur eines aktiven Knotens



EE: Execution Environment  
MEE: Management EE  
AA: Aktive Anwendung

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

11

## Aktive Anwendung (AA)

- Code, der in einem aktiven Knoten verarbeitet wird
  - interpretierter Quell-Code, z.B. Safe-TCL
  - Zwischencode, z.B. Java Byte Code
  - plattformabhängiger Binärcode
  - on-the-fly Übersetzung
- Herkunft
  - im aktiven Knoten bereitgestellte Bibliothek
  - Capsule
  - auf Verlangen heruntergeladen, z.B. durch Angabe einer URL im aktiven Paket → dynamische Erweiterbarkeit
- Ziele
  - Mobilität / Plattformunabhängigkeit
  - Sicherheit
  - Effizienz

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

12

## Programmierung

- Spezifikation skalarer Argumente
  - Auswahl vordefinierter Berechnungen
- Ereignis gesteuerte Berechnungen
  - Binden von Code an bestimmte Ereignisse
- universelle Programmiersprache
  - Beispiele: Java, Python
  - Problem: fehlender Ressourcenzugriff  
→ Erweiterung der virtuellen Maschinen
- spezielle Programmiersprache
  - Beispiele: PLAN, NetScript
  - Vorteil: erhöhte Sicherheit durch eingeschränkten Sprachumfang

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

13

## Ausführungsumgebung

- Execution Environment (EE)
- Umgebung zur Ausführung aktiver Anwendungen
  - AA Interpreter
  - sichere Sandbox
  - begrenzte Ressourcen
- mehrere EEs pro Knoten
- definiert virtuelle Maschine und stellt dem Code eine Programmierschnittstelle bereit
- Beispiele: ANTS, ALIEN
- Management EE überwacht und steuert aktiven Knoten
  - privilegierter Zugriff auf Systemressourcen
  - Überwachung und Steuerung der Hardware, NodeOS, EEs

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

14

## NodeOS

- Betriebssystem (operating system) des aktiven Knotens
- stellt API für AAs und EEs zur Verfügung
- begrenzt und verwaltet Systemressourcen, z.B.
  - CPU
  - Speicher
  - Threads
  - Kommunikationskanäle zum Senden/Empfangen aktiver Pakete
- Scheduling
- Authentifizierung und Autorisierung
- Accounting
- Beispiele: Linux, Nemesis, Scout

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

15

## Komponenten im Datenpfad

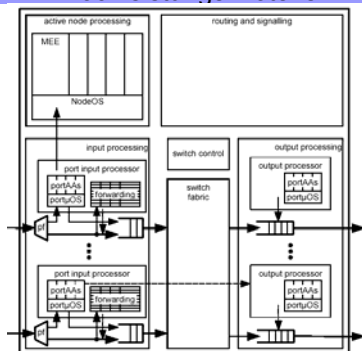
- Paketfilterung und -klassifikation
  - bestimmen, ob Pakete aktive Verarbeitung benötigen
  - schneller Datenpfad
  - aktiver Datenpfad
  - Kontrolldaten
- Schneller Datenpfad
  - darf nicht durch aktiven Pfad beeinträchtigt werden
  - Eingangsverarbeitung
    - Klassifikation
    - Bestimmung Ausgangs-Port
    - Header-Aktualisierung
  - Switch-Matrix
  - Ausgangsverarbeitung
    - Scheduling

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

16

## Architektur eines aktiven Hochleistungsknotens

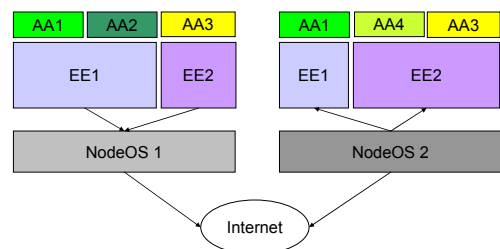


SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

17

## Multiplexing / Demultiplexing



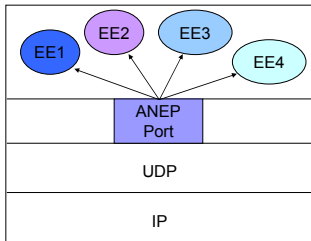
SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

18

## Active Network Encapsulation Protocol

- well-known UDP-Port für ANEP



SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

19

## ANEP Header-Format

Version	Flags	Type ID
ANEP Header Length		ANEP Packet Length
Optionen		
Nutzlast		

- Version = 1
- Flags zeigen Aktionen bei unbekanntem Typ an
  - MSB=0: Default-Weiterleitungsmechanismus
  - MSB=1: Löschen des Pakets
- ANEP Header Length inklusive Optionen
- TypeID: Umgebung zur Evaluation der Nachricht
- Optionen
  - Source ID: eindeutige ID des Senders, z.B. IP-/MAC-Adresse
  - Destination ID: eindeutige ID des endgültigen Ziels
  - Integrity Checksum: Prüfsumme über ganzes ANEP-Paket
  - Non-Negotiated Authentication: Einweg-Authentifizierung

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

20

## Beispiele aktiver Anwendungen

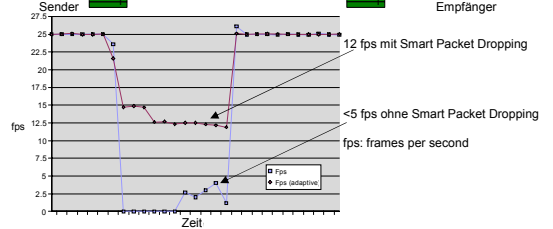
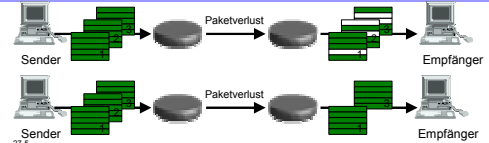
- aktive Staukontrolle
  - üblich: Router beginnt bei Überlast Pakete willlos wegzuerwerfen
  - Ansatz: intelligentes Wegwerfen von Paketen, z.B.
    - B- statt P-Frames bei MPEG
    - Wegwerfen von Komponenten höherer Frequenzen
    - Smart Packet Dropping
- nomadische Router
  - adaptive Router zwischen mobilem Endsystem und Netz
- Multicast
- Web-Caching
- Manipulation von Datenströmen
- intelligentes Routing
- Netz-Management und Monitoring

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

21

## Smart Packet Dropping



SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

22

## Multicast

Beispiel: Python Based Active Router (Baumgartner, 2002)

```
class SimpleMulticast (ARServiceHandler):
    def forward(self, pkt):
        # extract address list from first code block
        alist=cPickle.loads(pkt.cb(0))
        ifcs={}
        # scan addr.list and create if/addr.list structure
        for i in alist:
            interface=pad.queryRoute(i) ['if']
            if not ifcs.has_key(interface):
                ifcs[interface]=[]
            ifcs[interface].append(i)
        # scan if/addr. list structure and send packets
        for i in ifcs.keys():
            p=pad.Packet()
            p.cb(0)=cPickle.dumps(ifcs[i])
            p.cb(1)=pkt.cb(1)
            p.send({'dest':ifcs[i][0], 'ptype':pkt})
        return
```

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

23

## Web-Caching

- Verkehrsüberwachung in den Routern
- Organisation überlappender Multicast-Gruppen
- Austausch der Informationen zwischen den Routern
- Aufbau von URL-Tabellen
- Aktive Verarbeitung von HTTP-Requests
- Transparentes Überschreiben von IP-Adressen
- Auswahl des besten Servers in Abhängigkeit verschiedener Parameter, z.B. Netzlast, Hop-Anzahl,
- Unterstützung dynamischer Web-Seiten durch Speichern und Ausführen von Programmen

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

24

## Manipulation von Datenströmen

- Kodieren / Dekodieren (z.B. Vorwärtsfehlerkorrektur)
- Transformation
- Komprimieren / Dekomprimieren
- Verschlüsseln / Entschlüsseln



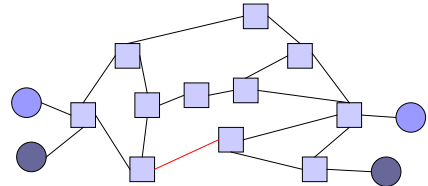
SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

25

## Intelligentes Routing

- Konventionelle Routing-Verfahren ignorieren Netzbelastung, Staus, verfügbare Bandbreiten, lokale Policies (z.B. IP Telefonie vs. ftp)
- Aktive Knoten können zusätzliche Informationen zum Routing nutzen, z.B. Auswahl von alternativen Wegen bei detektierten Staus
- Granularität: global, Flows, Pakete
- Multi-Path Routing



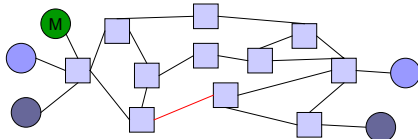
SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

26

## Aktives Management

- Heutige Netzmanagement-Mechanismen sind statisch u. wenig flexibel
  - Überwachung oft durch SNMP (MIB-Standardisierung)
  - Konfiguration über Command Line Interfaces (CLI)
- Aktives Netzmanagement → Scripts, Filtern u. Verarbeiten von Daten
- konventionell:
  - Überwachung durch Abfragen oder Alarme
  - Rekonfiguration als Reaktion⇒ > 2 RTTs
- Aktives Monitoring und Management
  - Aktiver Knoten kann nach Feststellen eines Ereignis sofort reagieren.



SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

27

## Sicherheit

- Nur berechtigte Benutzer sollen in der Lage sein, Code auf Netzelementen auszuführen.  
→ Authentifizierung und Autorisierung von Benutzern
- Denial-of-Service-Attacken  
→ Limitierung der Ressourcennutzung
- Schutz der aktiven Anwendungen untereinander  
→ Beschränkung der Instruktionsmengen und Adressräume bei Binärcode
- Anforderung an Compiler sicheren Code zu generieren  
→ Authentifizierung von Compilern
- Schutz der Capsules und Flow-Zustände vor Knoten  
→ Integrität

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

28

## Ende-zu-Ende Argument

- Saltzer, Reed, Clark: End-To-End Arguments in System Design, ACM Trans. on System Design, 1984
- Entwurfsprinzipien zur Platzierung von Funktionen in einem System
  1. Funktion (oder Dienst) sollte nur in einem Subsystem (z.B. einer Schicht) implementiert werden,
    - wenn sie vollständig darin implementiert werden kann, oder
    - wenn bei teilweiser Implementierung im Subsystem die Gesamtleistung eines Systems gesteigert werden kann.
  2. Nur Funktionen, die von allen Anwendungen benötigt werden, sollten im Netz realisiert werden.
- Ziel: Effizienz und einfache Struktur von Netzknoten

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

29

## Ende-zu-Ende Argument und Aktive Netze

- Stehen Aktive Netze im Widerspruch zum Ende zu Ende Argument ?
  - Programmierbarkeit scheint 2. Entwurfsprinzip zu widersprechen, da nicht alle programmierten Funktionen für alle Benutzer notwendig sind.
  - Programmierbarkeit erlaubt einem Netzbenutzer genau die benötigten Funktionen in einzelnen Netzknoten zu implementieren.

SS 02

Torsten Braun (Universität Bern): Netze und Protokolle für das Internet

30