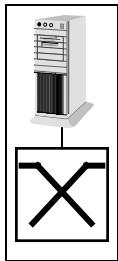


Netze und Protokolle für das Internet



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

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

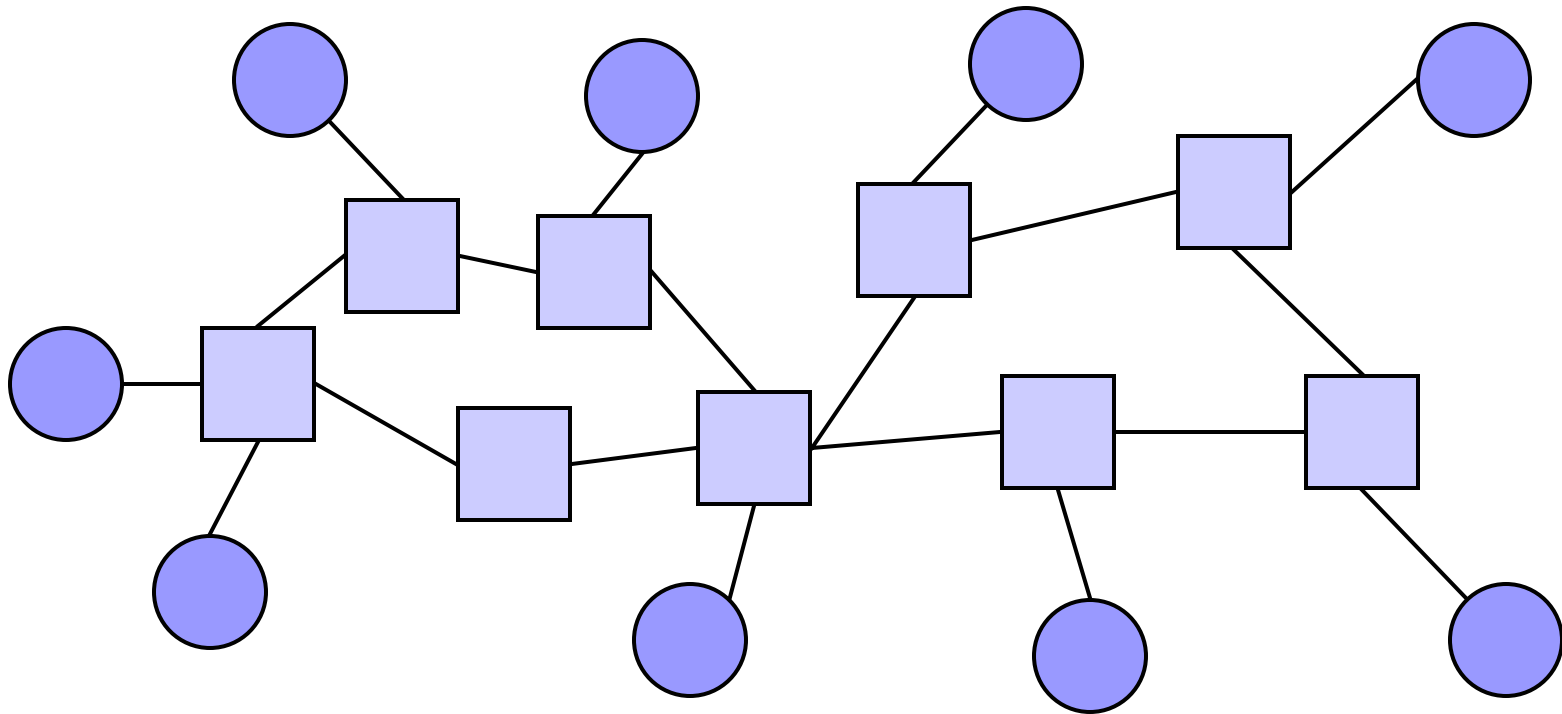
Einführen bzw. Testen

innovativer, individueller Dienste

⇒ programmierbare Netzelemente

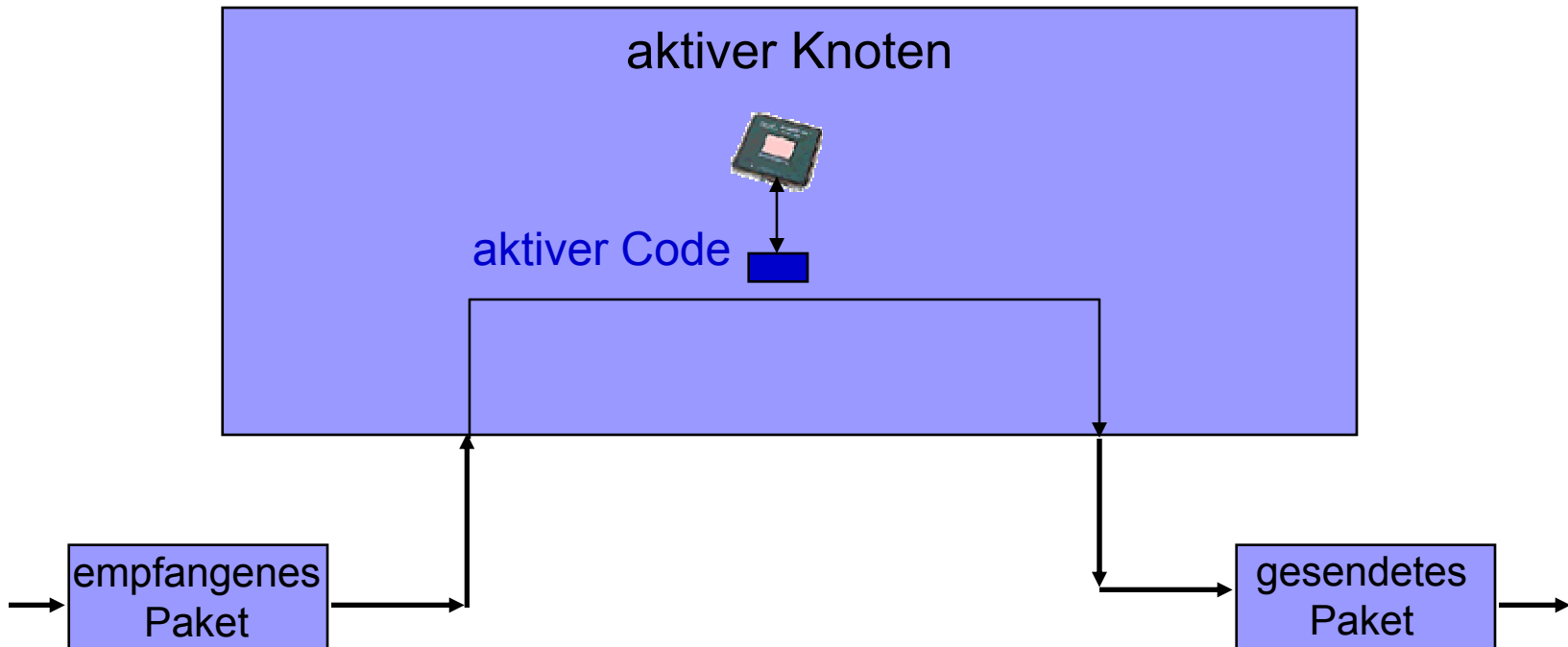
Passive Networking

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



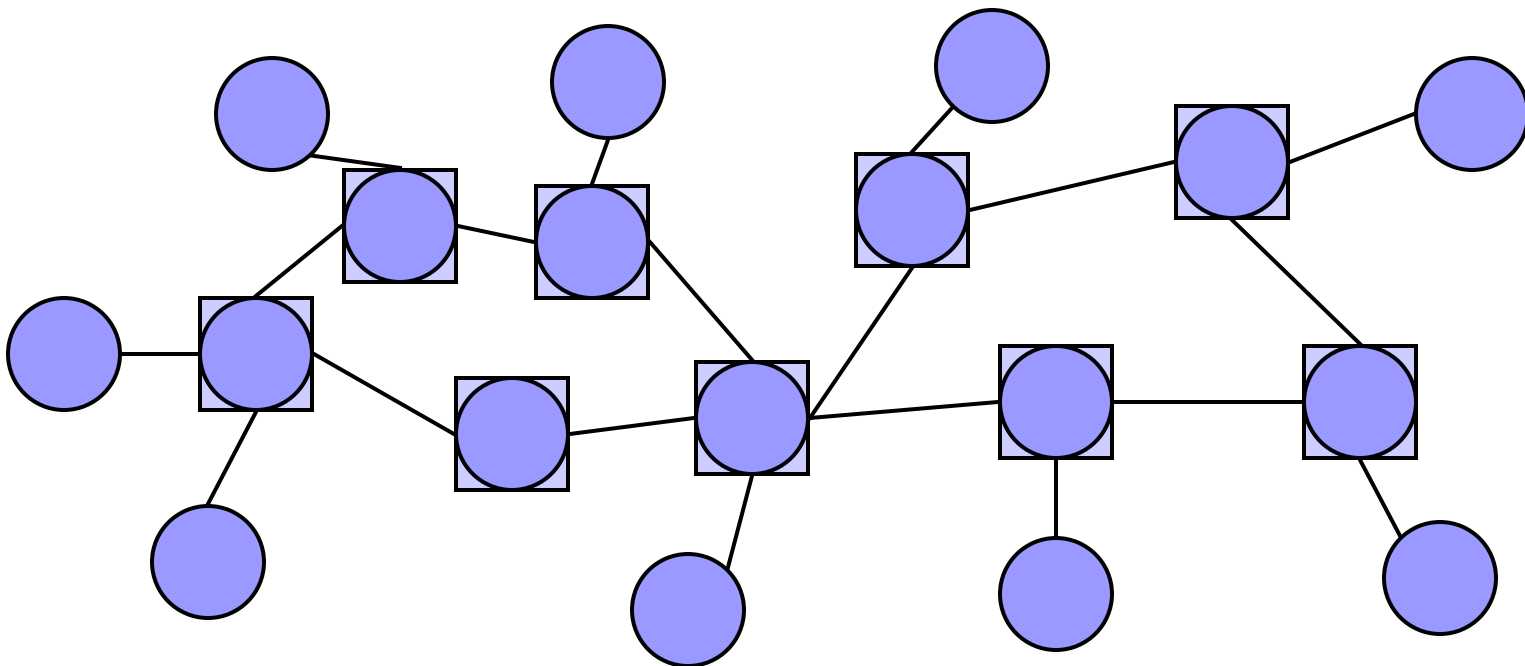
Active Networking

- Verarbeitung
 - store
 - compute
 - forward



Modell Aktiver Netze

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



Active Networking Typen

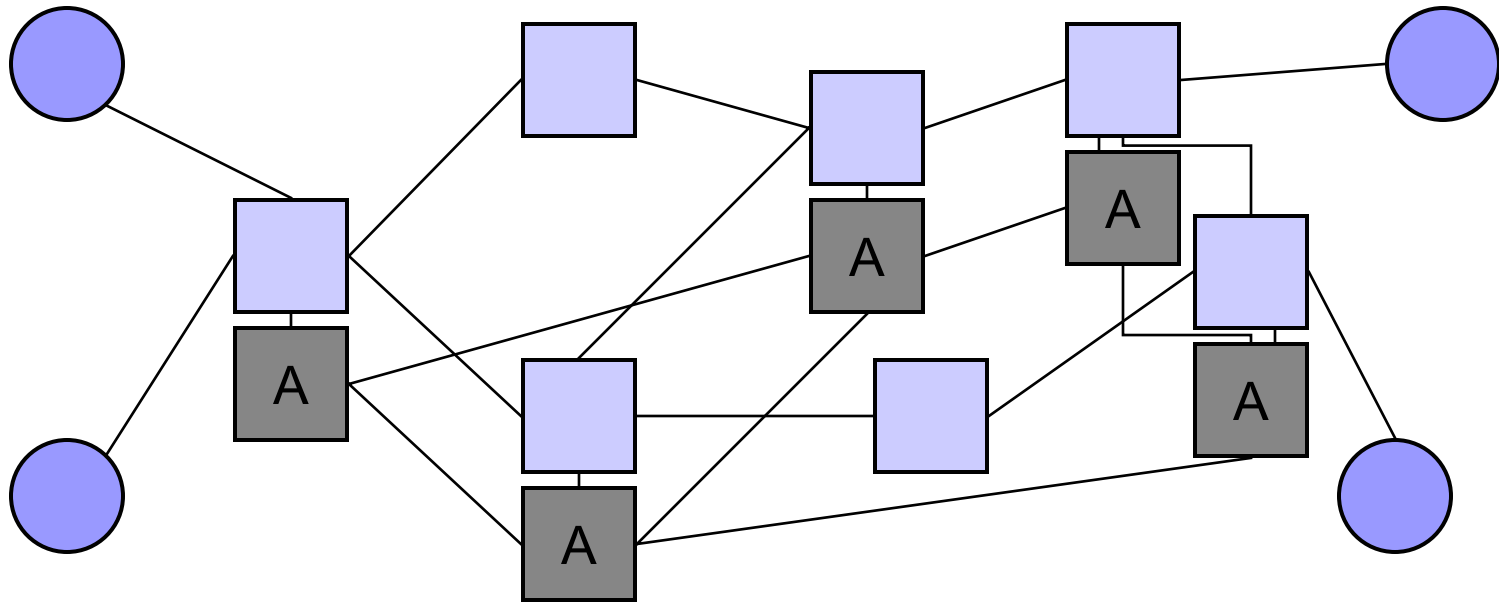
▪ moderat

- Code wird durch Service Provider zur Verfügung gestellt.
 - Beispiel: Intelligente Netze
- Benutzer selektieren verfügbaren Code.
- erlaubt die schnelle Einführung neuer Protokolle und Dienste

▪ intensiv

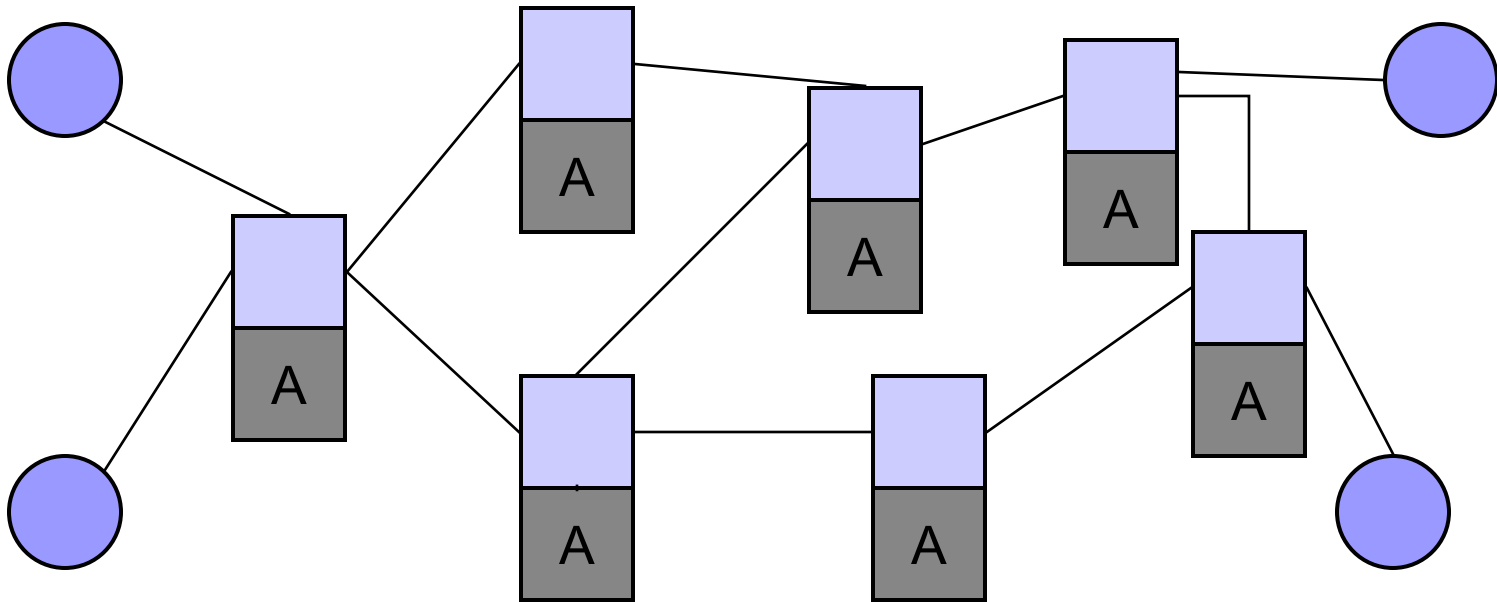
- Benutzer injiziert eigenen Code in aktiven (Daten-)Paketen (Capsules).
- Knoten führt Capsules in Execution Environment (EE) aus.

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

Eingebettete Intelligenz

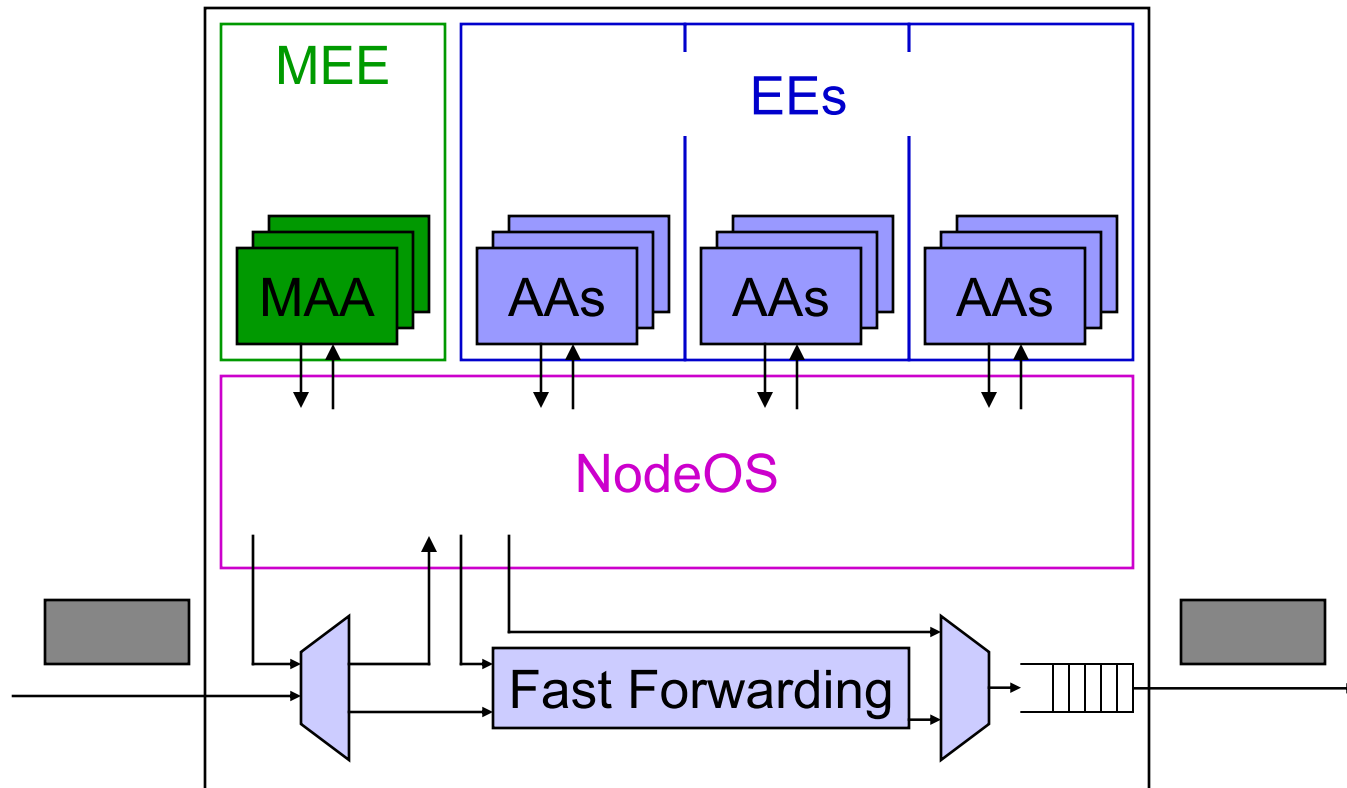


- eingebettete Intelligenz in ausgewählten (oder allen) Netzelementen
- Einrichtung der Dienste kann in-band erfolgen.
- erlaubt inkrementelles und selektives Einführen von Diensten

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

Architektur eines aktiven Knotens



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

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

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

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

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

Komponenten im Datenpfad

- **Paketfilterung und -klassifikation**

bestimmen, ob Pakete aktive Verarbeitung benötigen

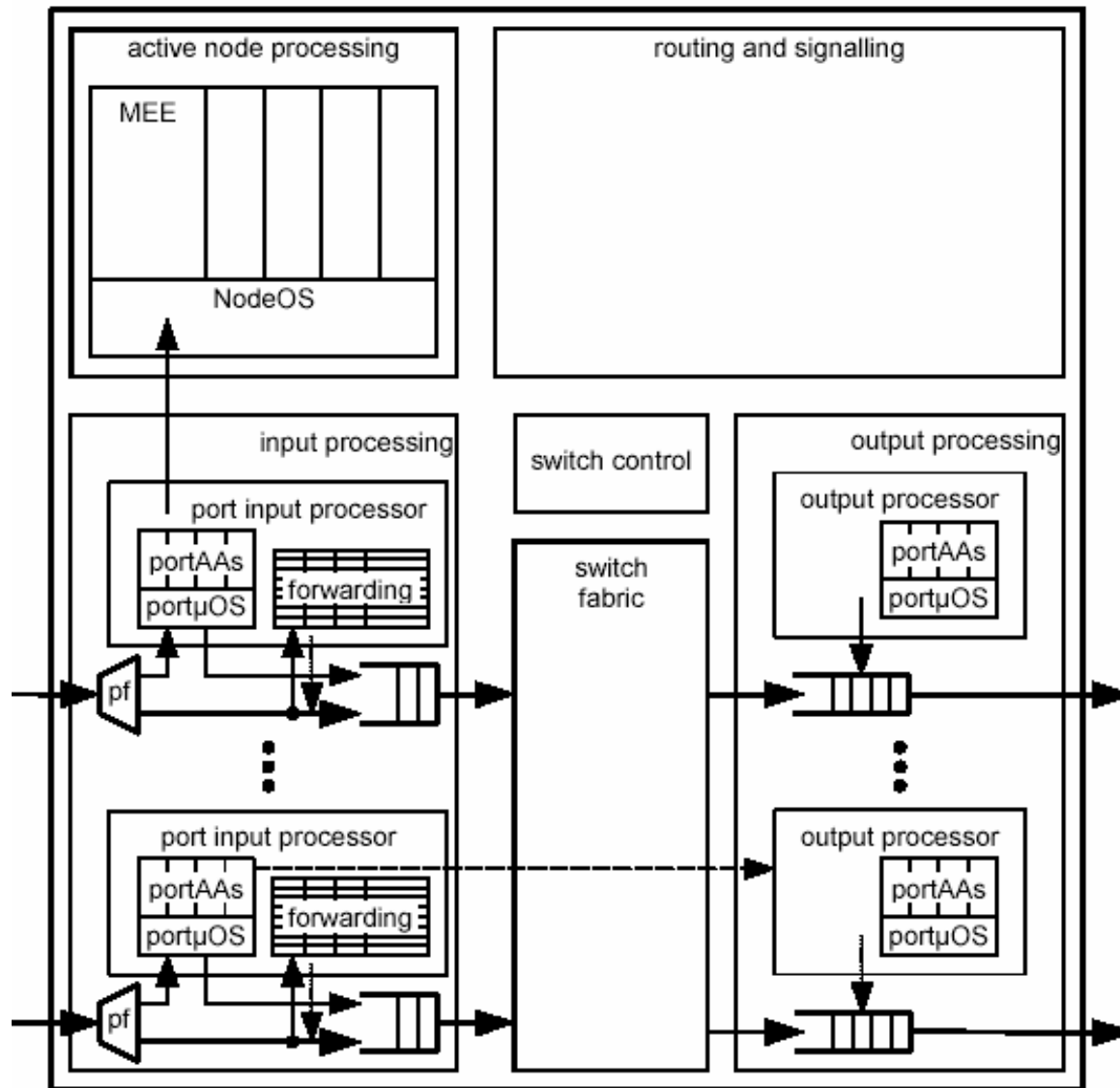
- schneller Datenpfad
- aktiver Datenpfad
- Kontrolldaten

- **Schneller Datenpfad**

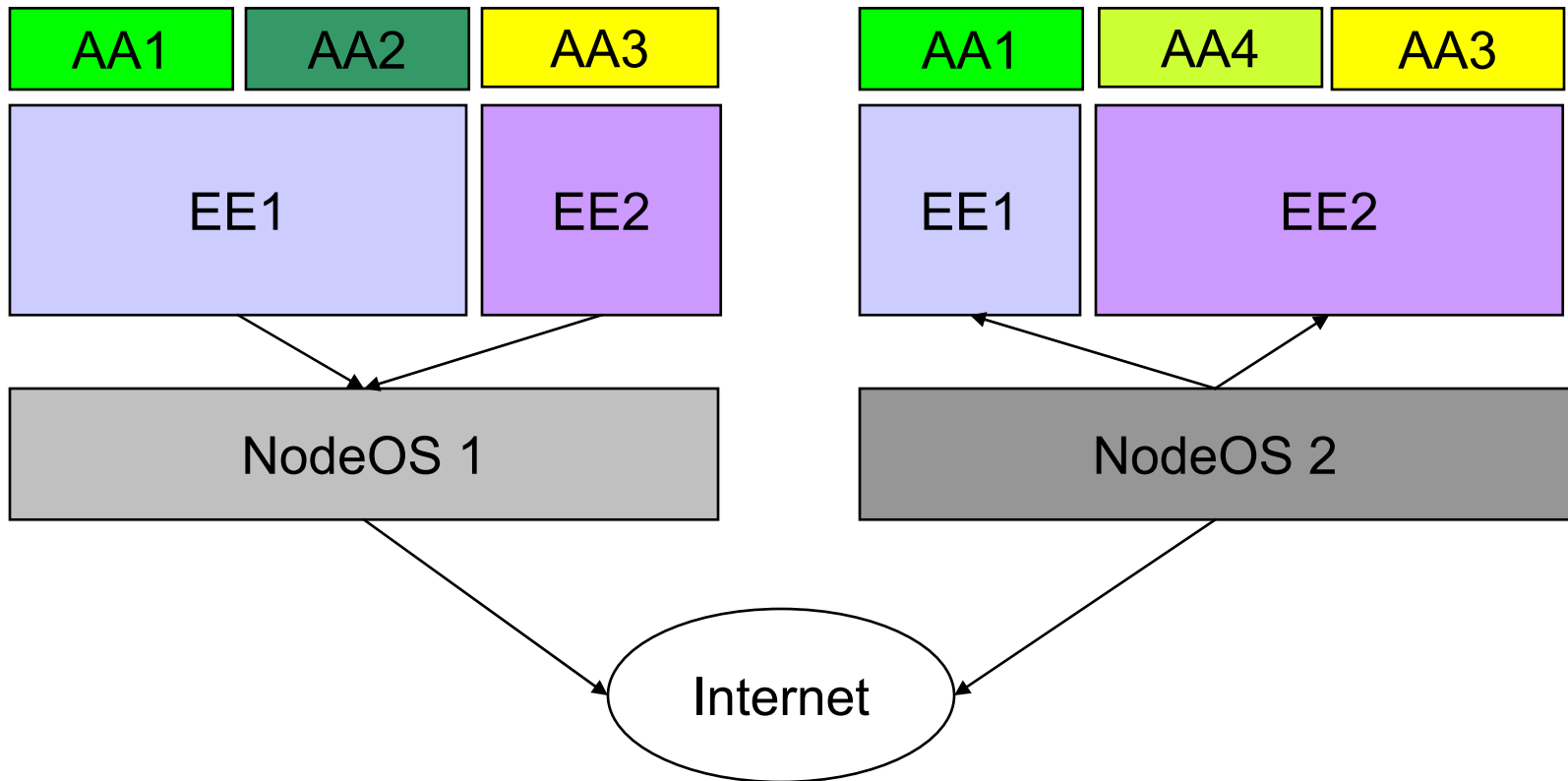
darf nicht durch aktiven Pfad beeinträchtigt werden

- Eingangsverarbeitung
 - Klassifikation
 - Bestimmung Ausgangs-Port
 - Header-Aktualisierung
- Switch-Matrix
- Ausgangsverarbeitung
 - Scheduling

Architektur eines aktiven Hochleistungsknotens

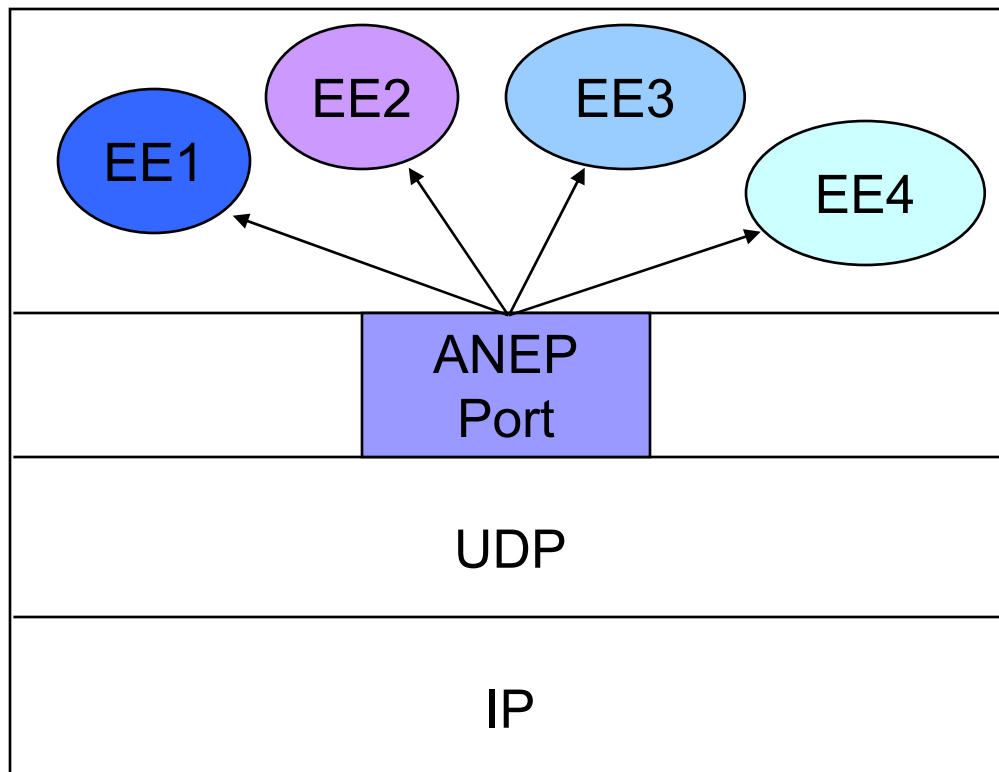


Multiplexing / Demultiplexing



Active Network Encapsulation Protocol

- well-known UDP-Port für ANEP



ANEP Header-Format

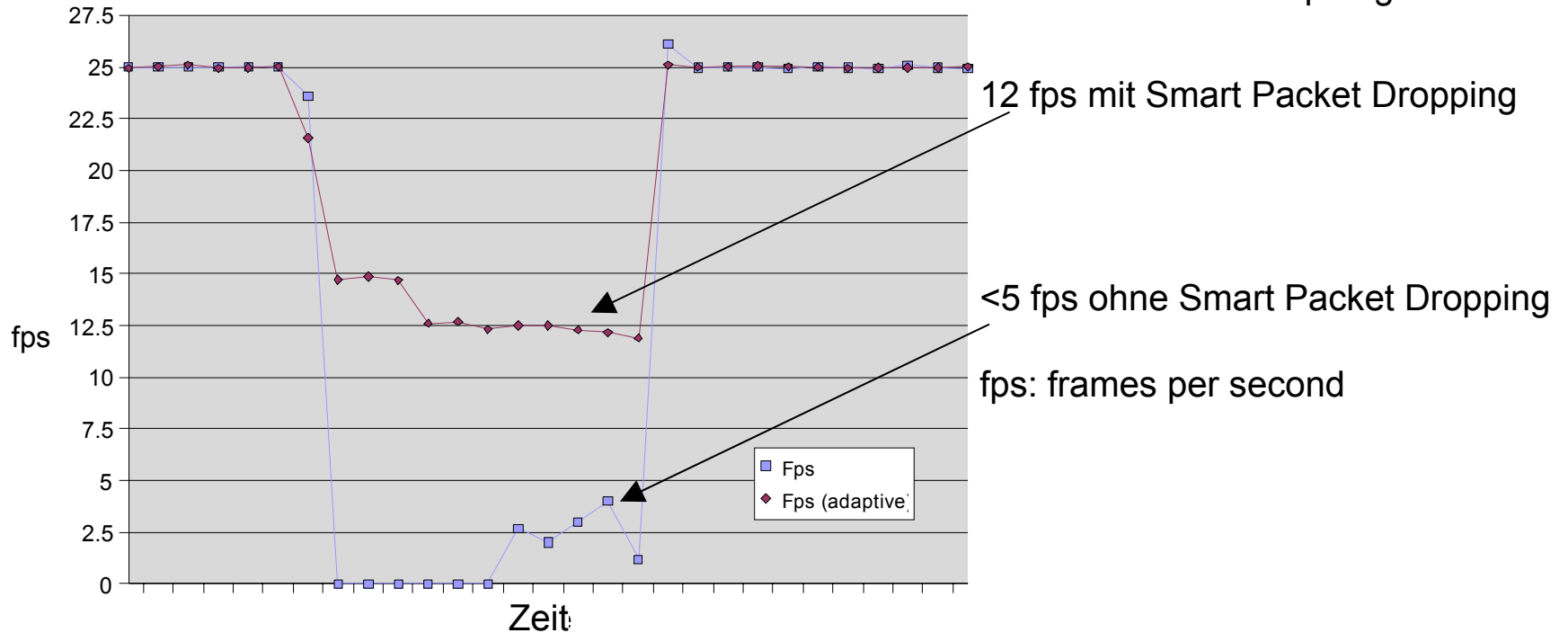
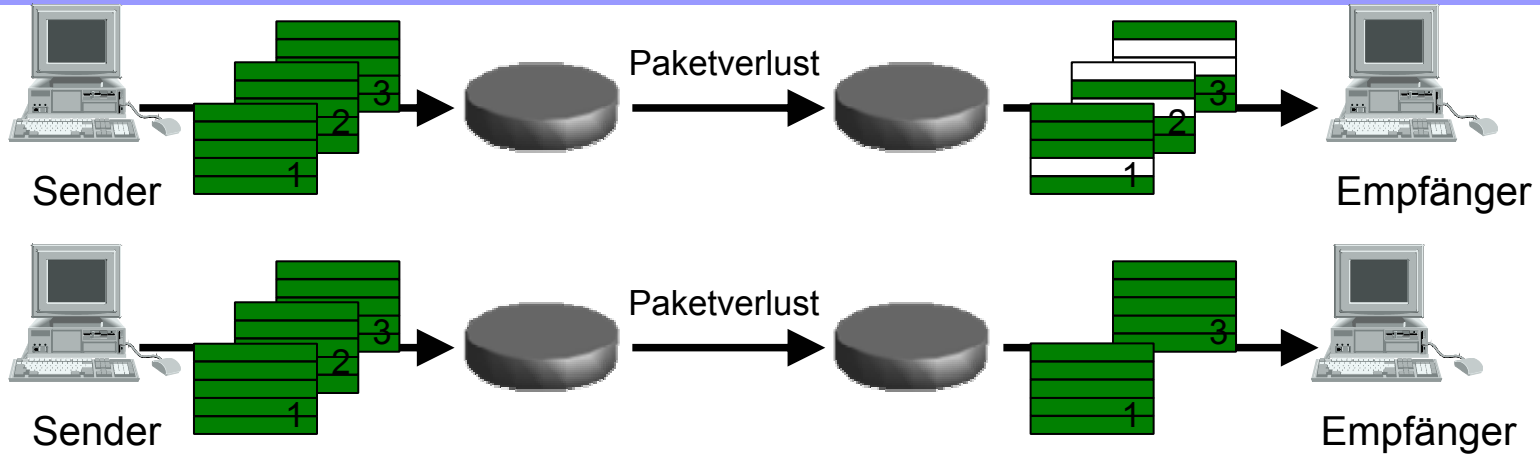
Version	Flags	Type ID
ANEP Header Length		ANEP Packet Length
Optionen		
Nutzlast		

- 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

Beispiele aktiver Anwendungen

- aktive Staukontrolle
 - üblich: Router beginnt bei Überlast Pakete wahllos wegzuwerfen
 - 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

Smart Packet Dropping



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
```

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

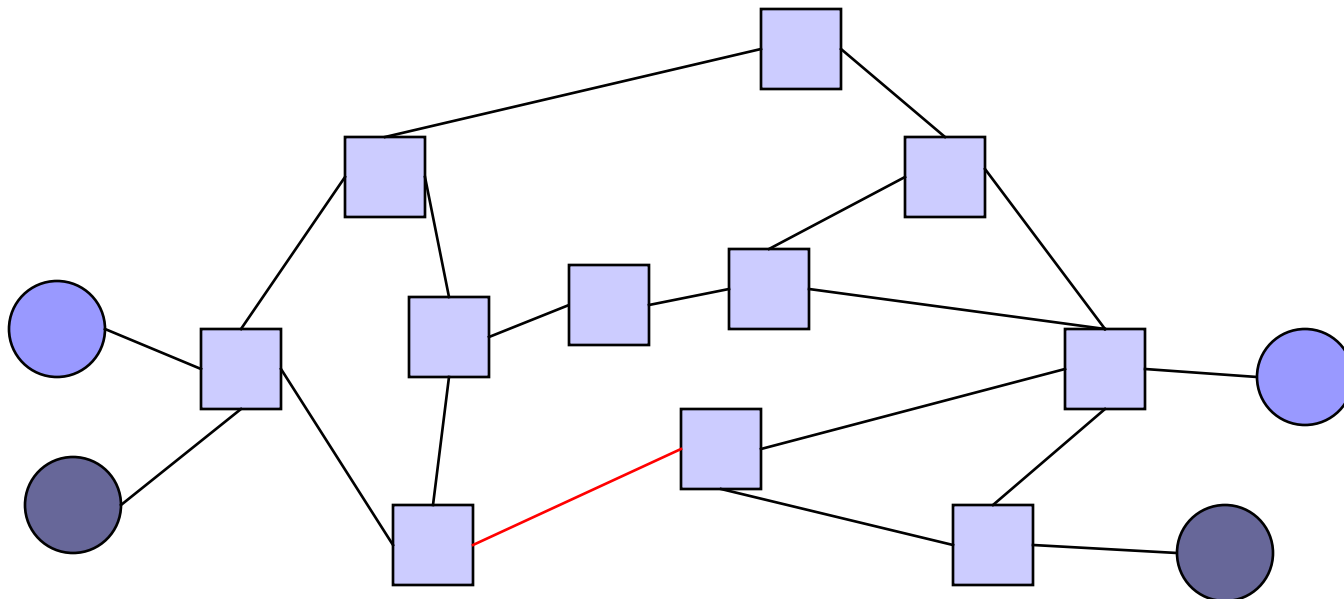
Manipulation von Datenströmen

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



Intelligentes Routing

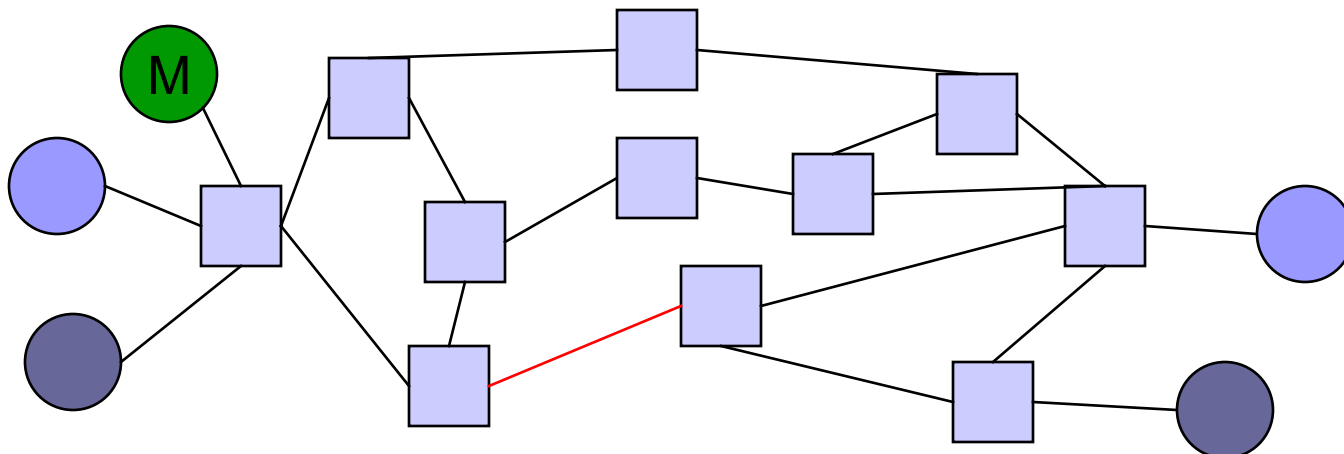
- Konventionelle Routing-Verfahren ignorieren Netzbelastung, Staus, verfügbare Bandbreiten, lokale Policies (z.B. IP Telefonie vs. ftp)
- Aktive Knoten können zusätzliche Informationen zum Routing nutzen, z.B. Auswahl von alternativen Wegen bei detektierten Staus
- Granularität: global, Flows, Pakete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



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.



Sicherheit

- Nur berechnigte 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

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

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.