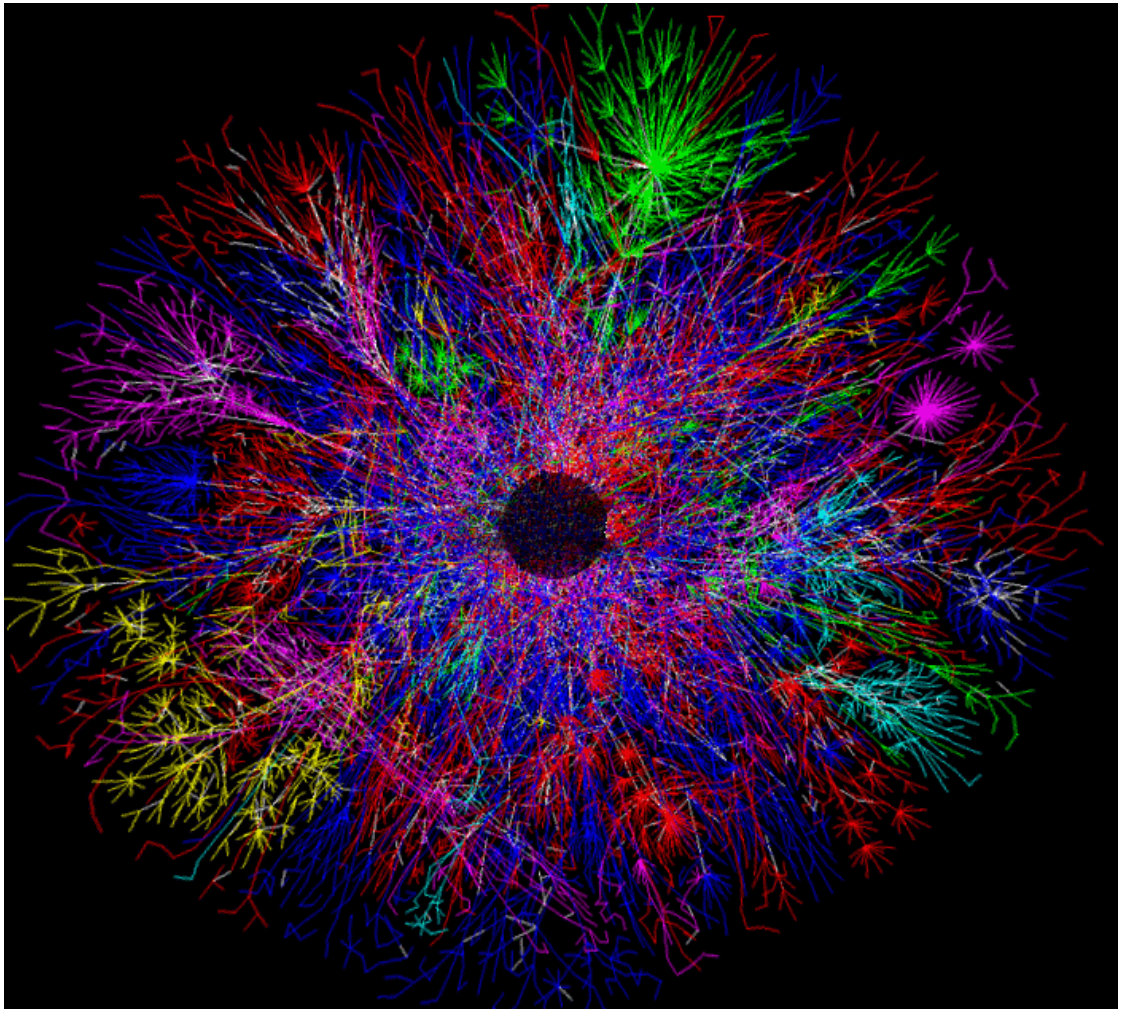
Distack: Framework zur einfachen Entwicklung von Mechanismen zur lokalen und verteilten Angriffserkennung

Christoph P. Mayer und Thomas Gamer

EWNS2008

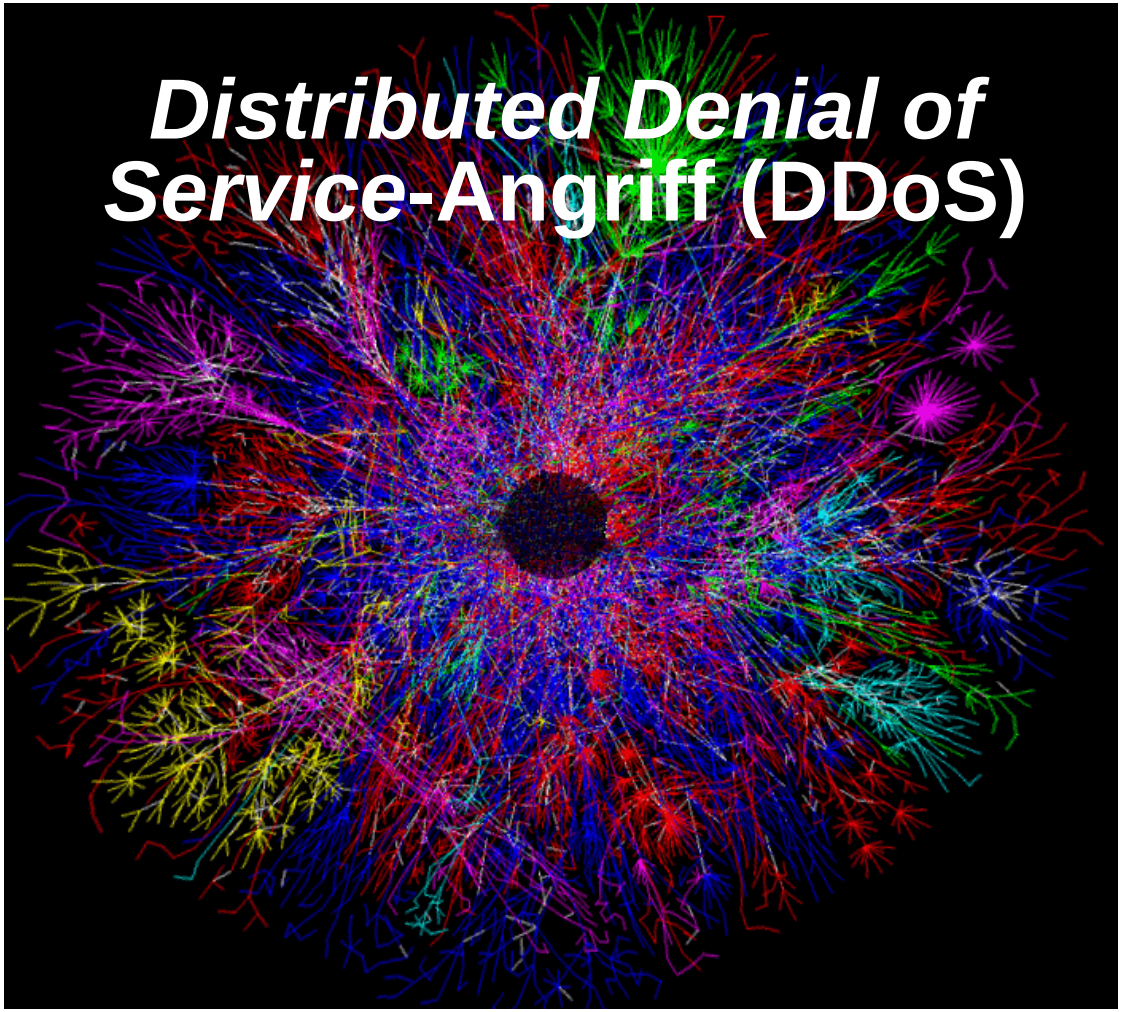
Institut für Telematik

Was ist das?



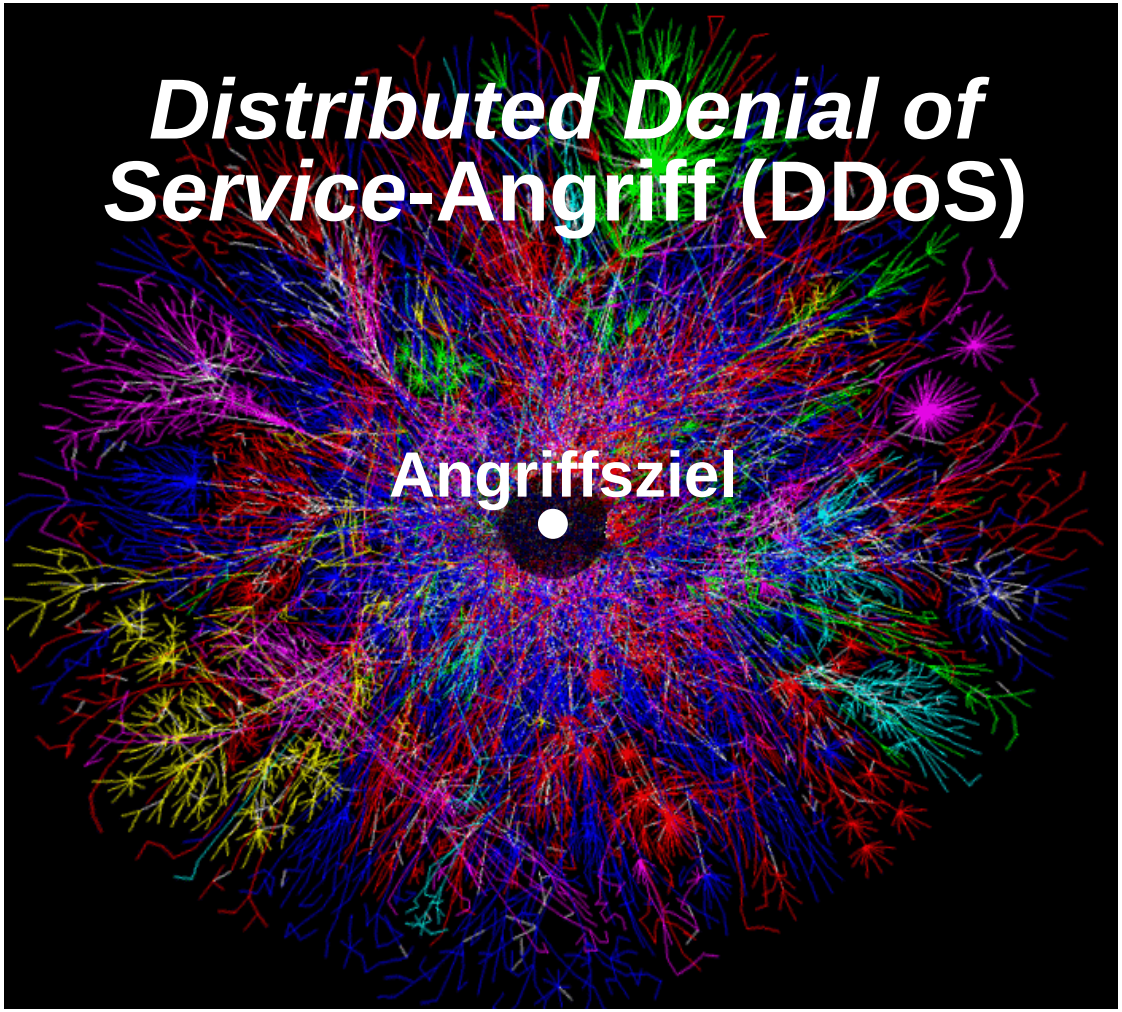
Quelle: Prolexic

Was ist das?



Quelle: Prolexic

Was ist das?

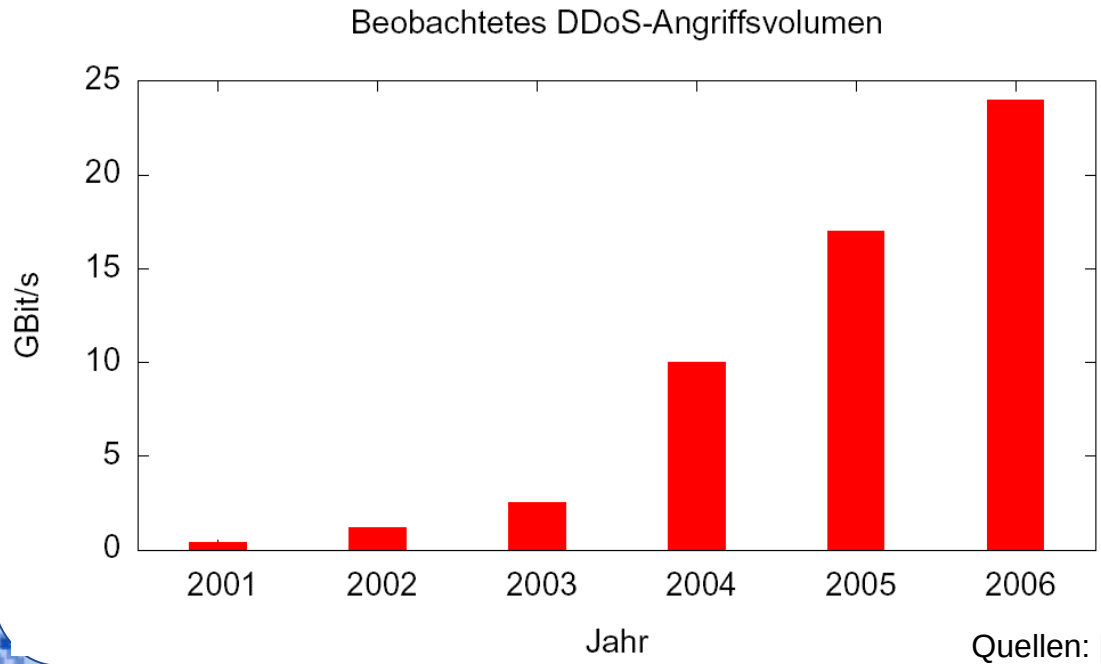


Quelle: Prolexic

Motivation

„Neuseeländer soll Botnetz mit 1,3 Millionen Computern gesteuert haben“ (Heise-Online, Nov. 2007)

„DDoS und Wurmangriffe größte Gefahren, die zur Zeit das Internet bedrohen“ (Worldwide Infrastructure Security Report, Arbor Networks, 2007)

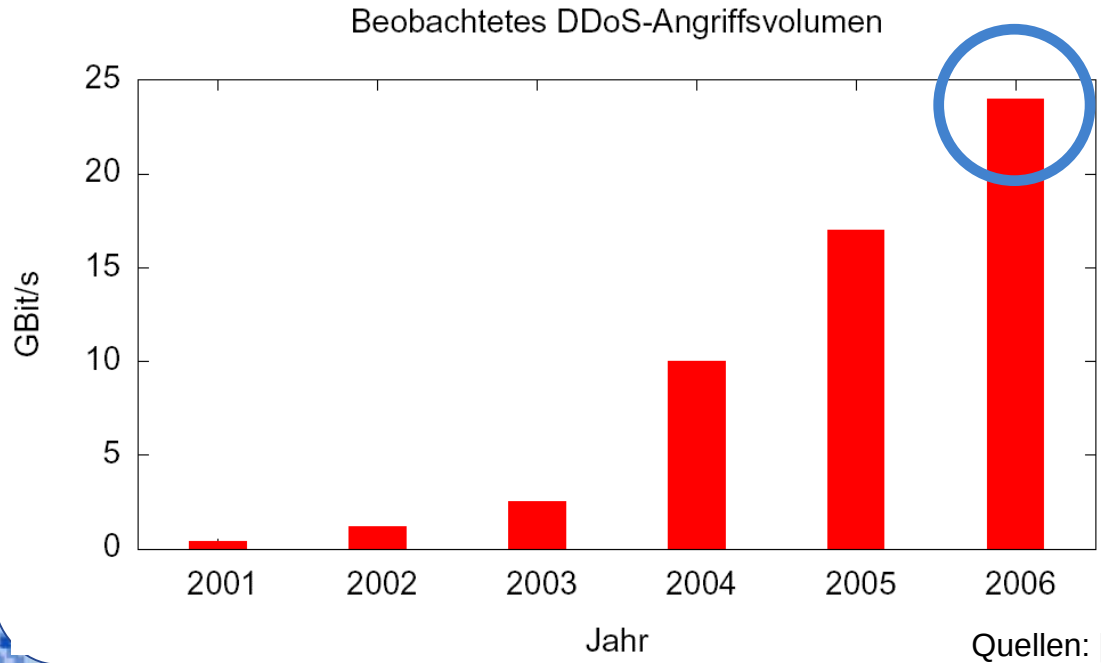


Quellen: [Daten: Arbor Networks], [Bild, Wikipedia]

Motivation

„Neuseeländer soll Botnetz mit 1,3 Millionen Computern gesteuert haben“ (Heise-Online, Nov. 2007)

„DDoS und Wurmangriffe größte Gefahren, die zur Zeit das Internet bedrohen“ (Worldwide Infrastructure Security Report, Arbor Networks, 2007)



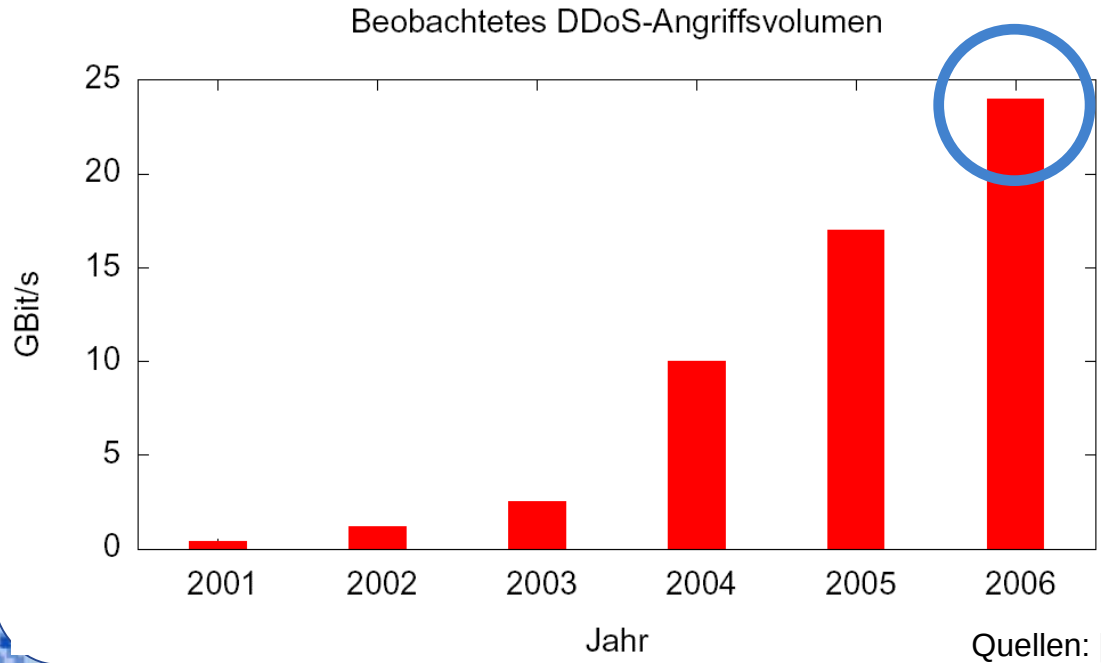
1,3 Millionen Systeme senden Ø 19 KBit/s

Quellen: [Daten: Arbor Networks], [Bild, Wikipedia]

Motivation

„Neuseeländer soll Botnetz mit 1,3 Millionen Computern gesteuert haben“ (Heise-Online, Nov. 2007)

„DDoS und Wurmangriffe größte Gefahren, die zur Zeit das Internet bedrohen“ (Worldwide Infrastructure Security Report, Arbor Networks, 2007)



1,3 Millionen Systeme senden Ø 19 KBit/s

ISA V.34 Modem, 1995

Quellen: [Daten: Arbor Networks], [Bild, Wikipedia]

Wo liegt das Problem?

- Technische Versiertheit der Angreifer
  - professionelle Entwicklung u. Verbreitung von Bots
  - intelligente Organisation und Koordination der Bots
    - P2P-artige Organisation des *Storm Botnet* (2007)
- Motivation der Angreifer
  - früher: Angriffe persönlich motiviert (Ruhm)
  - heute: finanziell oder politisch motiviert
- Unglaubliche Größe heutiger Botnetze

Vermietung von Botnetzen

→ Stunden-/tageweise Vermietung von Botnetzen

Schädigung von Mitbewerbern

→ Für Verkäufer wie Amazon.com ist Nichterreichbarkeit ein Desaster

Wo liegt das Problem?

- Technische Verspieltheit der Angreifer

Storm Botnet, 2007

- Verbreitet durch Email (*Storm Worm)*
  - Geschätzte infizierte Systeme 1-50 Millionen
  - P2P Organization und Koordination
- Unglaubliche Leistung
  - Mächtig genug um Staaten vom Internet abzuschneiden
  - Mehr Rechnpower als die besten Supercomputer
- Aktive Verteidigung des Botnet durch Besitzer
  - Analysen des Botnets aktiv verhindert

"The owners of the Storm botnet, whose identities are as yet unknown, could be preparing to sell off the 'services' of segments of the network, [...]" CNET News

- Für Verkäufer wie Amazon.com ist Nichterreichbarkeit ein Desaster

Wo liegt *wirklich* das Problem?

- Beispielhafte Probleme
  - Wenig Wissen über globales Verhalten von DDoS-Angriffen und Wurmausbreitungen
  - Fehlende verteilte kooperative Angriffserkennung und kooperative Gegenmaßnahmen
  - Kaum wiederverwendbare Ergebnisse, Arbeiten bauen wenig aufeinander auf

Grundproblem:

Komplexe Entwicklung und Evaluierung von Mechanismen zur lokalen und verteilten Verkehrsanalyse und Angriffserkennung

- Beispielhafte Hürden für Entwicklung und Evaluierung
  - Initialer Aufwand (z.B. Protokoll-Parser, Paket-Management, …)
  - Laufzeitumgebungen/Betriebssysteme (PCs, Router, Simulator, …)
  - Datenquellen (Tracedatei, Live-Verkehr, simulierter Verkehr, …)
  - Kommunikation zwischen entfernten Instanzen

Vorhandene Systeme

- Real World: Snort, PreludeIDS
  - Signatur-basiert (für Snort auch Spade anomaly preprocessor)
  - Kein Sampling möglich und nicht sinnvoll
  - Keine Granularität / intelligentes Laden von Signaturen/Modulen
  - Sensor/Manager, aber keine verteilte kooperative Angriffserkennung

- Forschung: Bro
  - Event-basiert (`con_attempt`, `con_finished`, ...)
    - Ungeeignet um Aggregate zu Überwachen
  - Kein Sampling, keine Granularität
  - Keine verteilten Instanzen

→ Laufzeitumgebungen? Simulator?

Unser Ansatz: Distack

Distack: Framework zur Angriffserkennung

distack

distributed anomaly-based attack detection

Hauptziele

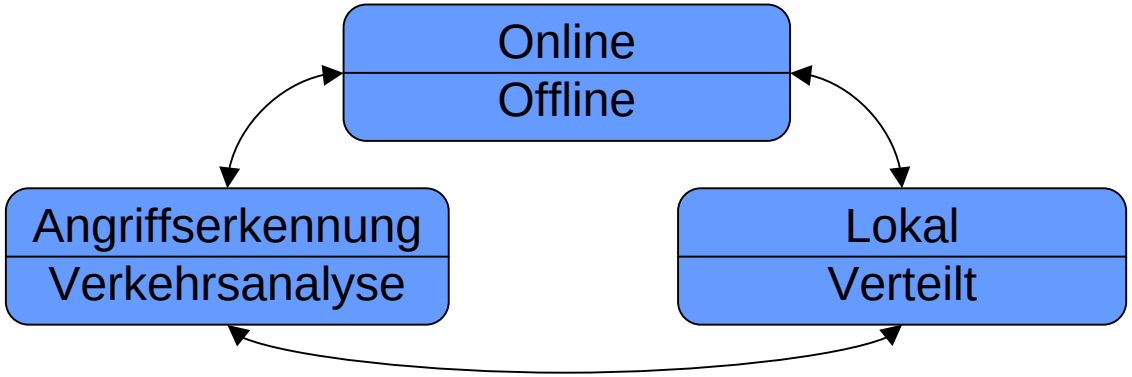
- Konzentration auf Methoden und Analysen selbst
- *Write once run everywhere*: Methoden transparent in unterschiedlichen Umgebungen betreiben
- Wiederverwendbarkeit durch Bausteine
- Bestmögliche Unterstützung für verteilte Angriffserkennung

Anforderungen

- Skalierbarkeit
  - Betrieb im Netzinneren auf Vermittlungssystemen
  - Hohe Verkehrslast, geringe Ressourcen
- Erweiterbarkeit
  - Neue Protokolle
  - Neue Methoden zur Angriffserkennung
- Flexibilität
  - Verschiedene Laufzeitumgebungen
  - Baukastenprinzip
- Robustheit
  - Vermeidung von Überlast
  - Protokoll-Parser

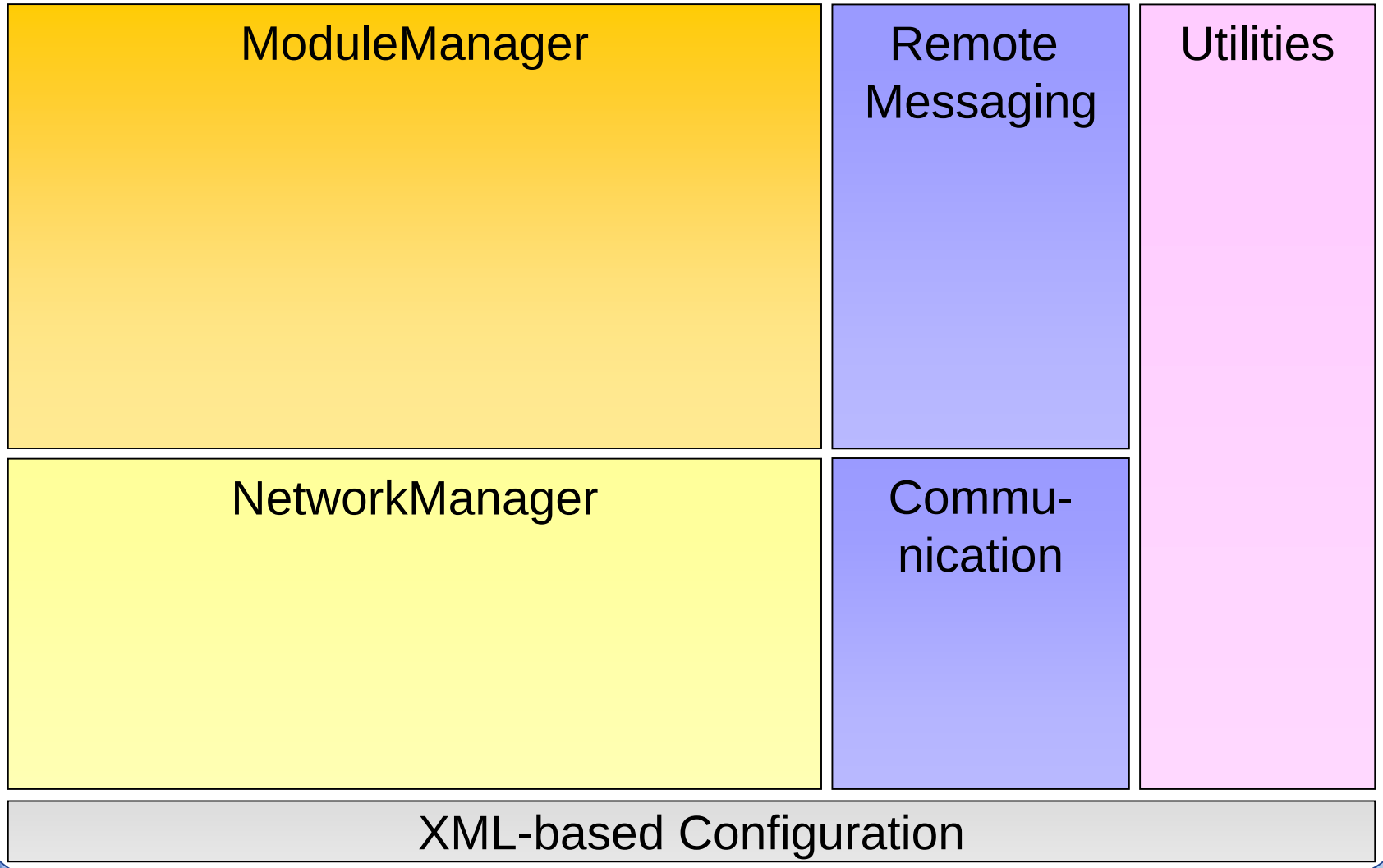
Ziel

- Flexibles Framework zur verteilten Angriffserkennung
  - Baustein-basiert (*Module*), flexibel konfigurierbar

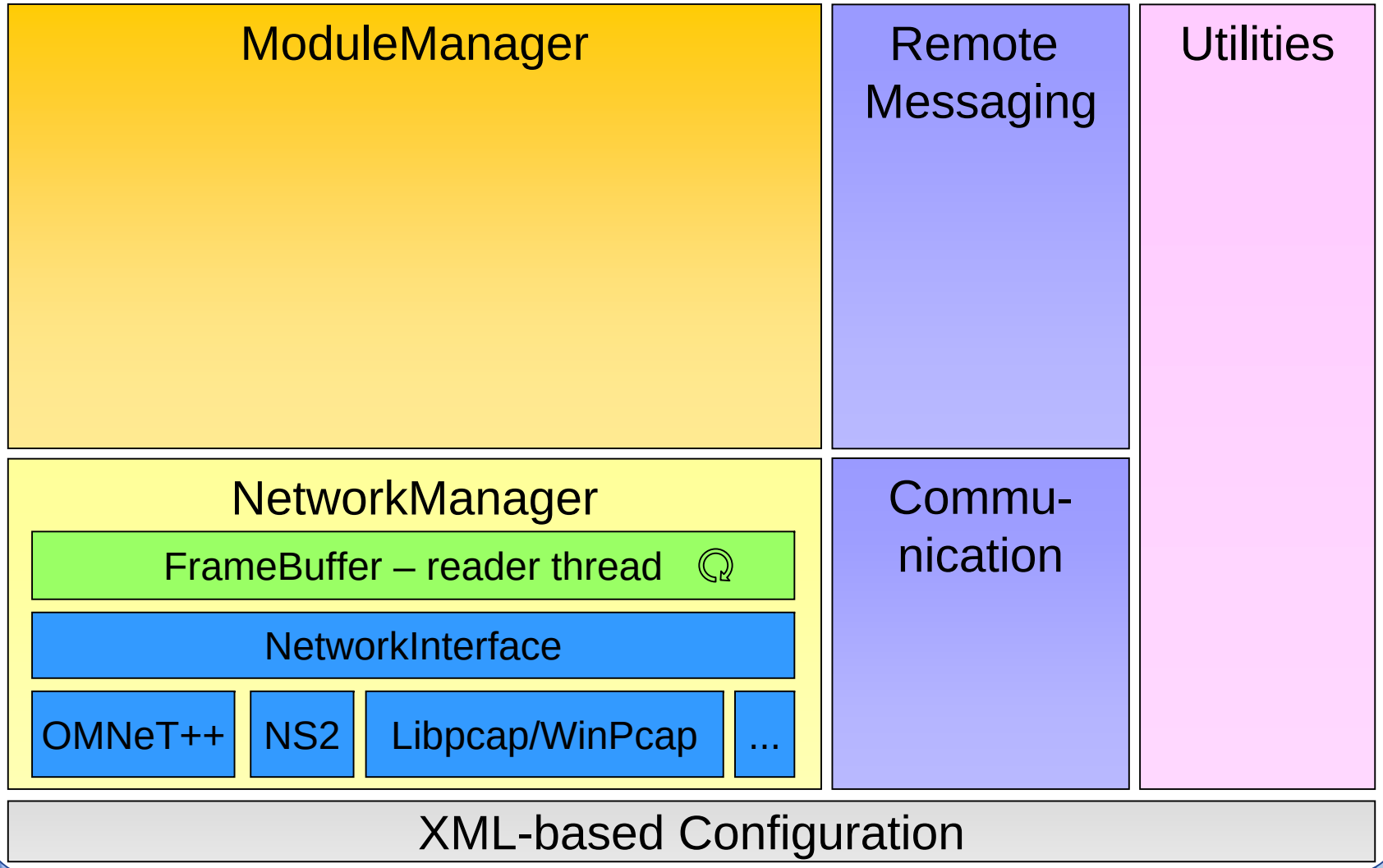


- Einfache Schnittstellen: Integration von Erweiterungen
  - Neue Protokolle, Methoden zur Angriffserkennung
  - Neue Komponenten (GUI, ...)
- Verschiedene Laufzeitumgebungen
  - Router, PC, Simulator, Netzwerkprozessor, ...
  - Laufzeitumgebung transparent für Module

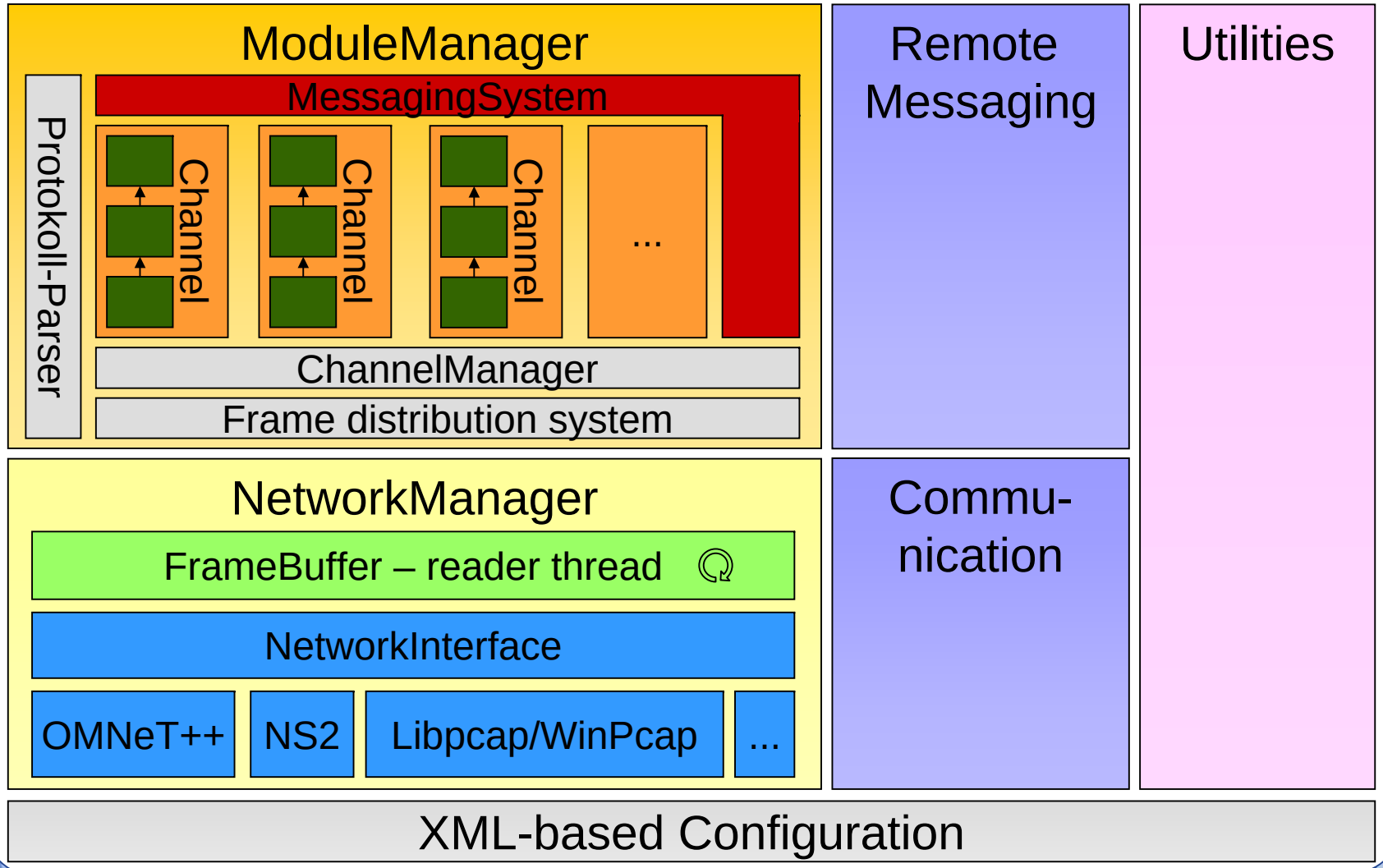
Architektur



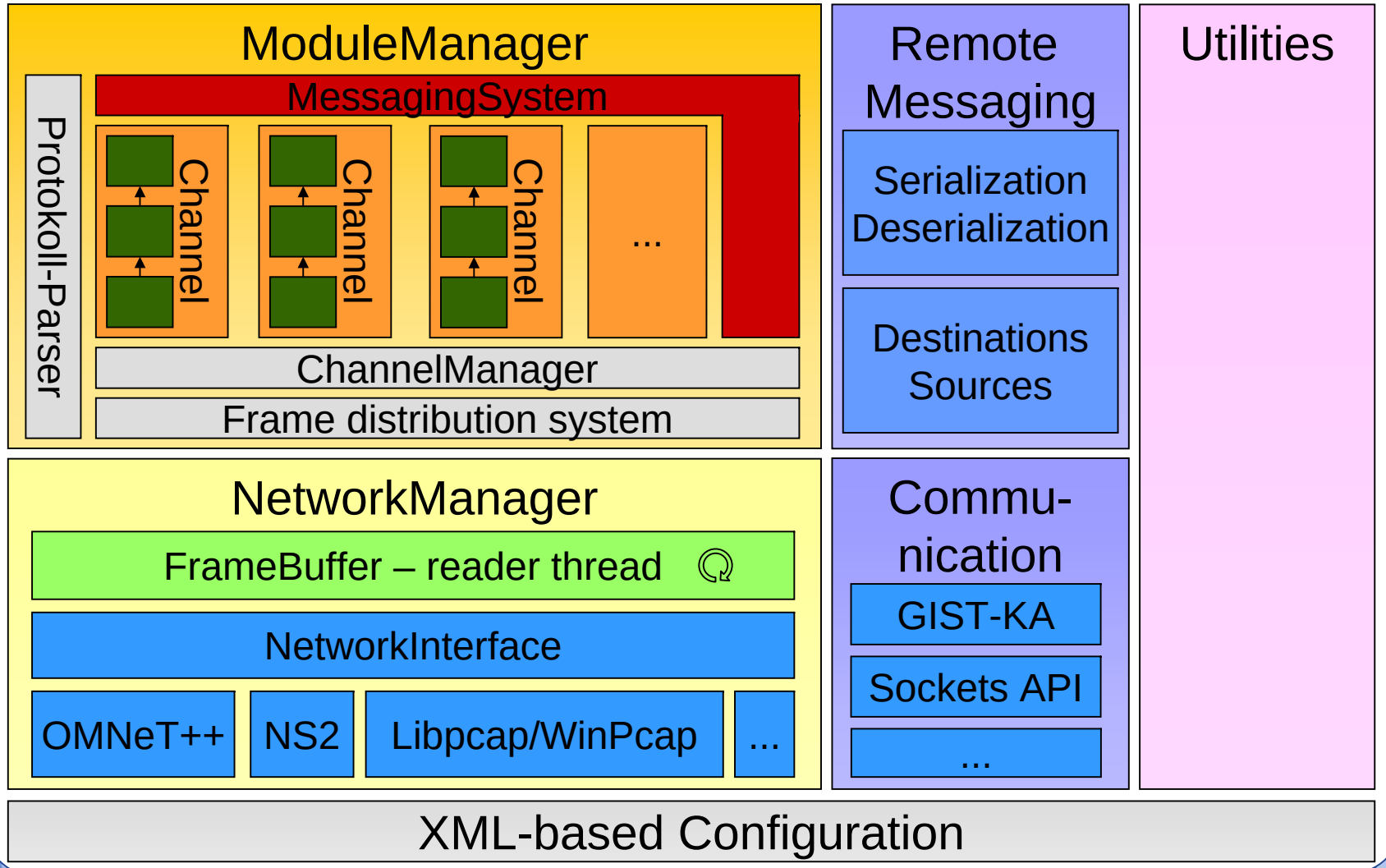
Architektur



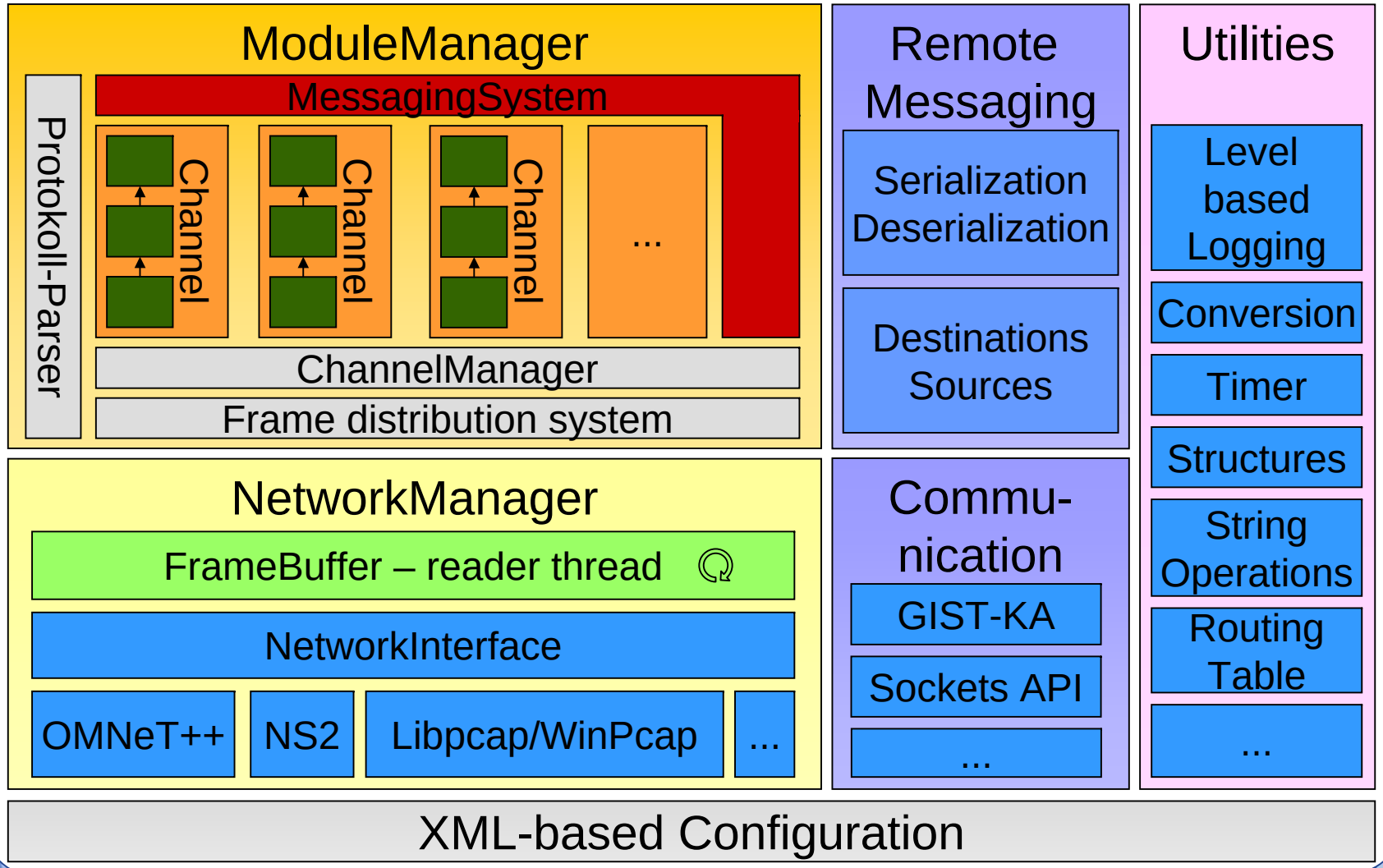
Architektur



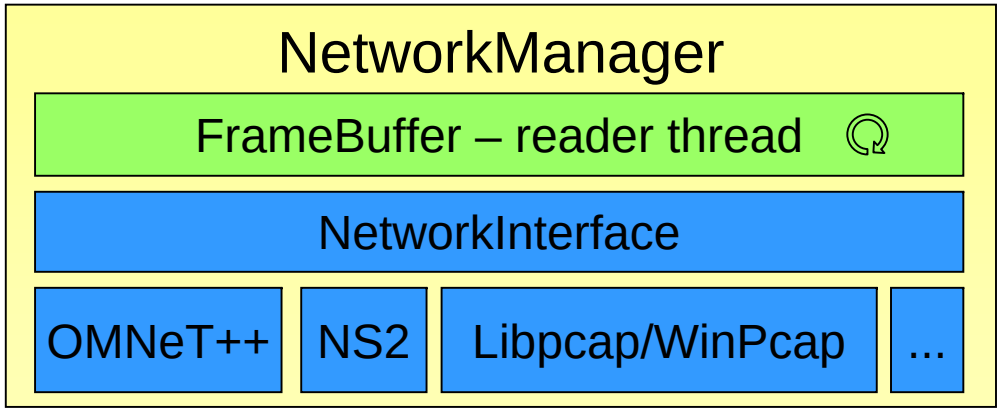
Architektur



Architektur

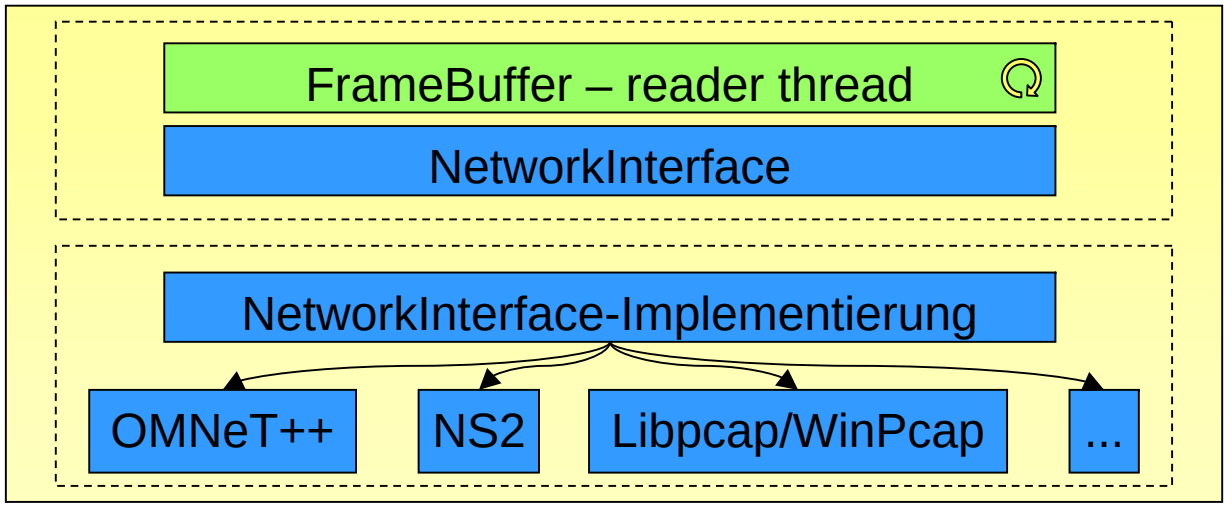


Architektur – Netzwerkabstraktion

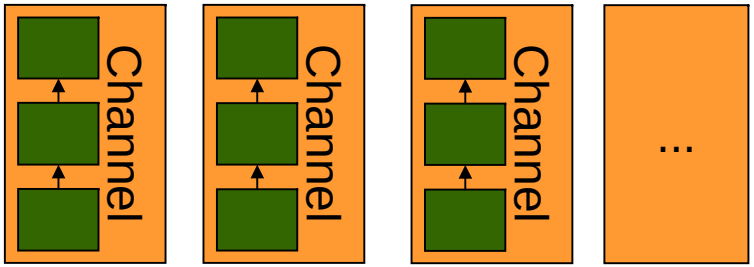


Netzwerkabstraktion

- Abstraktion des unterliegenden Netzes
  - Paketformate, notwendige Konvertierung
  - Abhängig von der Laufzeitumgebung
- Abfangen von Bursts
  - Zwischenspeicher für Frames
  - Begrenzungsmechanismen



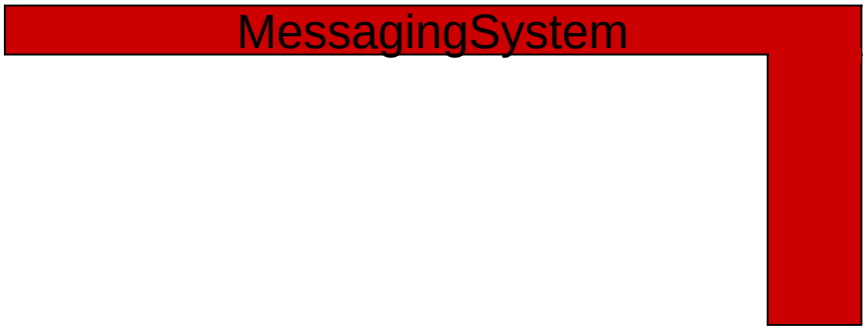
Architektur – Module



Module – funktionale Einheiten

- Modularisierung: Module als Bausteine
  - Klein und leichtgewichtig
  - In dynamische Bibliotheken ausgelagert
- Modul-Funktionalität
  - Sampling/Filtering/Überwachung von Frames
  - Sammeln von statistischen Daten, Analyse
  - Synchronisierung, grafische Ausgabe, Plotter
- Channels
  - Funktionale Kette von Modulen
  - Verarbeitung von Netzwerkframes
  - Gruppierung von zusammengehörigen Modulen
  - Granularitätsstufen

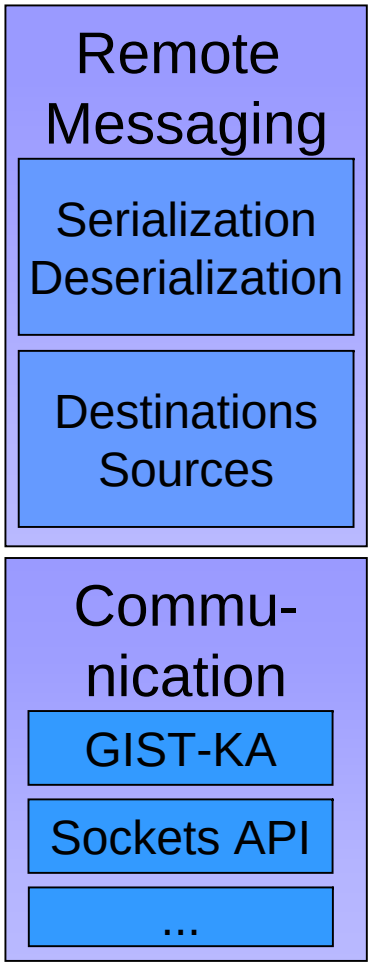
Architektur – Lokale Kommunikation



Lokale Kommunikation

- Datenzentrierte Kommunikation
  - Kommunikation zwischen Modulen
  - Informationsquellen nicht bekannt
  - Daten werden in das System gesendet
  - Synchrone/asynchrone Nachrichtenzustellung
- Registrierungs-basiert
  - Modul registriert sich für interessante Nachrichten
- Beispiel: Statistische Verteilung
  - Analyse-Modul sammelt Daten zu einer statistischen Verteilung
  - Sendet diese Verteilung periodische aus
  - Interessierte Module registrieren sich und erhalten die Nachricht

Architektur – Verteilte Kommunikation



Verteilte Kommunikation

- Kommunikation zwischen entfernten Modulen
  - Transparenter Nachrichtenversand und Empfang
  - Beliebig komplexe Nachrichtenobjekte
- Serialisierung
  - Nachrichtenobjekte serialisieren und deserialisieren
  - Keine Paketformate notwendig
- Adressierung durch Versandoptionen
  - Lokal, entfernt, lokal + entfernt
  - Empfängerliste, Nachbarn
- Datenübertragung
  - Austauschbare Kommunikationsschicht
  - GIST, Sockets-API, ...

Architektur – Konfiguration

XML-based Configuration

XML-basierte Konfiguration

- Allgemein
  - Netzwerk, Tracedatei, Speedup, ...
- Baustein-Konfiguration
  - Instanziierung und Konfiguration von Modul-Bibliotheken
  - Mehrfache Instanziierung einer Bibliothek
- Baukastenprinzip
  - Verschaltung von Modulen zu Channels
  - Channel definiert zusammengehörige Funktionalität
  - Weitere Gruppierung von Channels
    - Granularitätsstufen

XML-basierte Konfiguration – Beispiel

Externe Bibliothek

Lokaler Name

```
<module name="Modules">
  <submodule lib  ="ModuleSamplingSystematicCountBased"
             name ="SamplingItemOne">
    <configitem name="SamplingCount">200</configitem>
    <configitem name="SelectionCount">1</configitem>
  </submodule>
  ...
</module>
```

Modul-Konfiguration

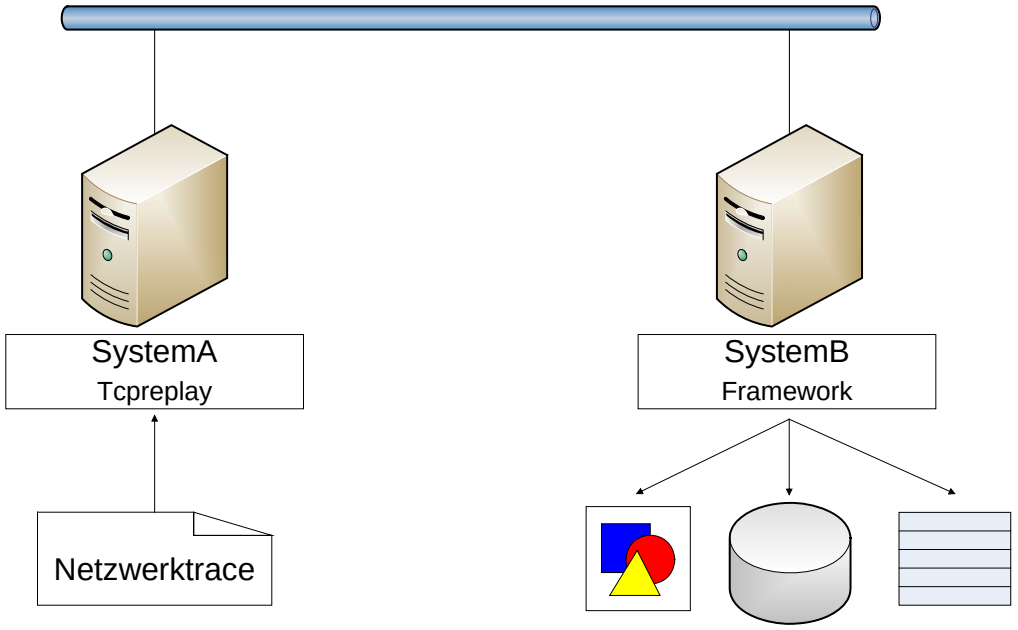
Channel Name

Channel Gruppe

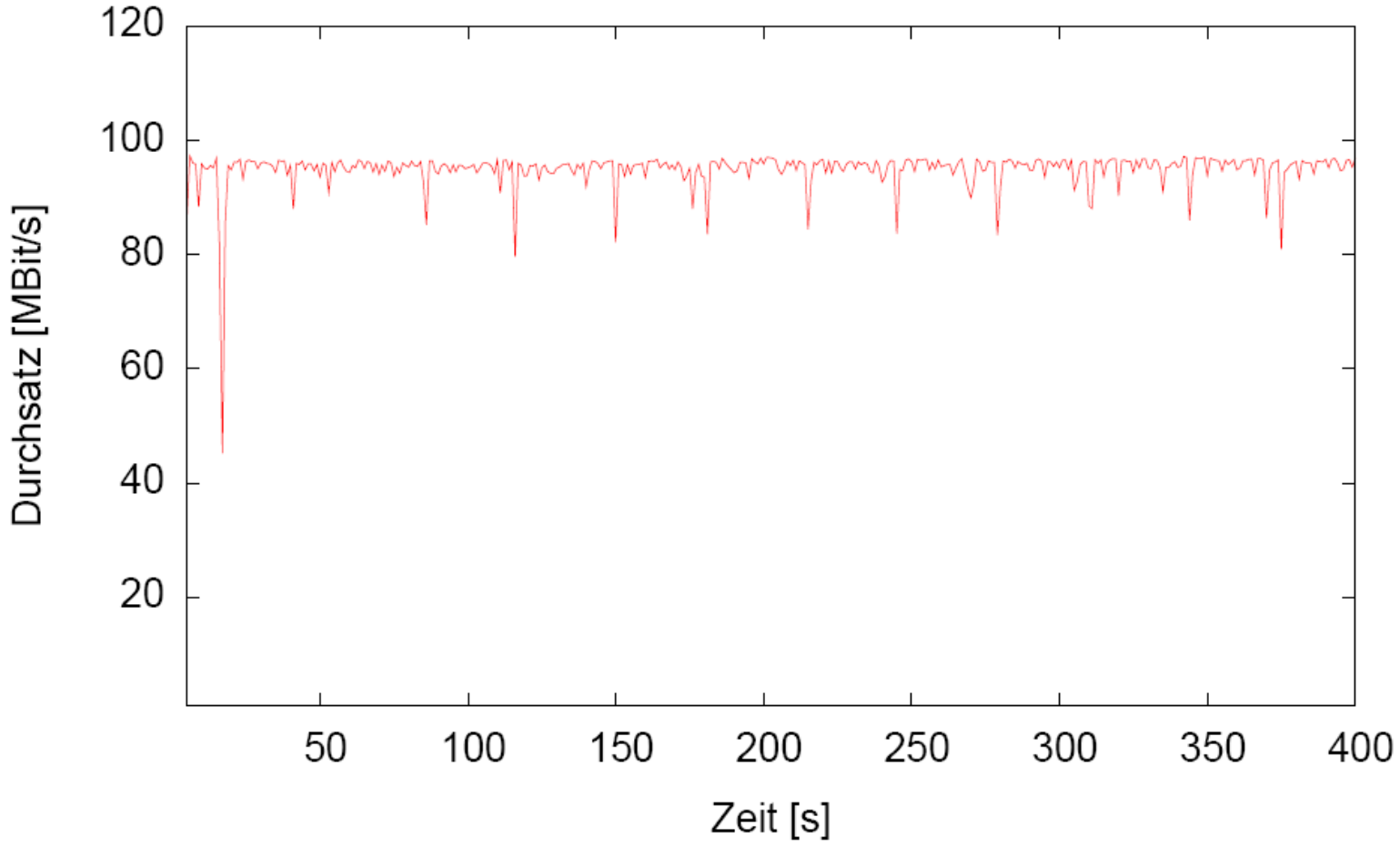
```
<module name="Channels">
  <submodule name="StageOne" stage="1">
    <configitem name="1">SamplingItemOne</configitem>
    <configitem name="2">...</configitem>
  </submodule>
  ...
</module>
```

Evaluation – Skalierbarkeit

- Systemaufbau
  - SystemA: Abspielen einer Netzwerktrace
  - SystemB: Betreibt Framework, Basisfunktionalität
  - Realistischer Verkehr, 100 MBit/s Ethernet

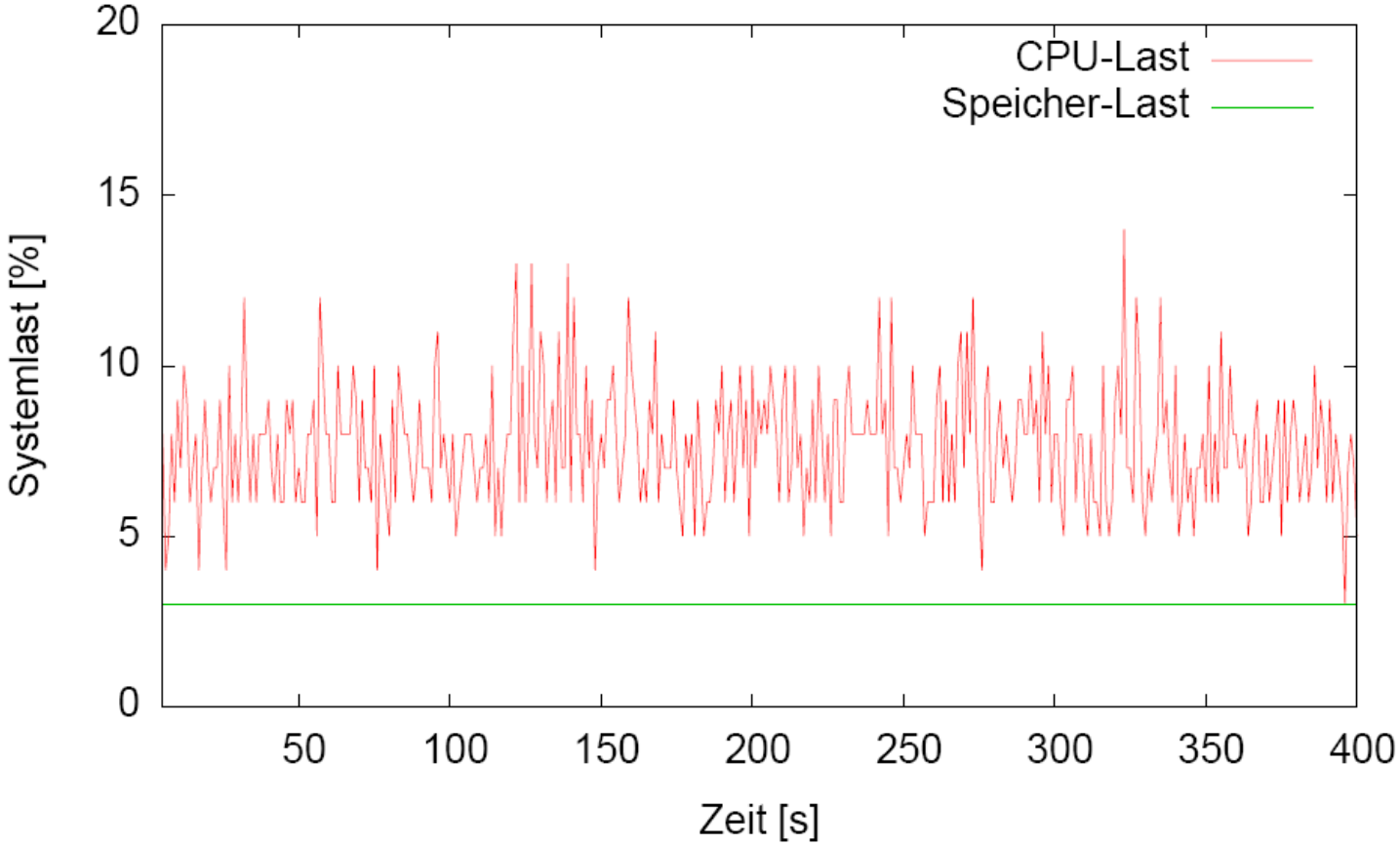


Evaluation – Skalierbarkeit



- Verarbeitung von 100 MBit/s kein Problem

Evaluation – Skalierbarkeit (2)



- CPU- und Speicherlast akzeptabel
- Bisher keine Optimierungen betrachtet

Evaluation – Flexibilität

- OMNeT++ als Laufzeitumgebung
  - Kapselung der `main`-Funktion als `cSimpleModule`
  - Framework als Bibliothek von OMNeT++ geladen
  - Netwerkabstraktion für OMNeT++
- Stolpersteine
  - Threads: bekommen keine Prozessorzeit
  - Timer: Zeitdomäne der Simulationszeit
  - Pakete: andere Darstellung in OMNeT++
  - …
- Resultat
  - Lokale und verteilte Angriffserkennung in großen Netzen
  - Simulation von DDoS- und Wurmangriffen

Evaluation – Flexibilität

- OMNeT++ als Laufzeitumgebung

Christoph P. Mayer und Thomas Gamer,

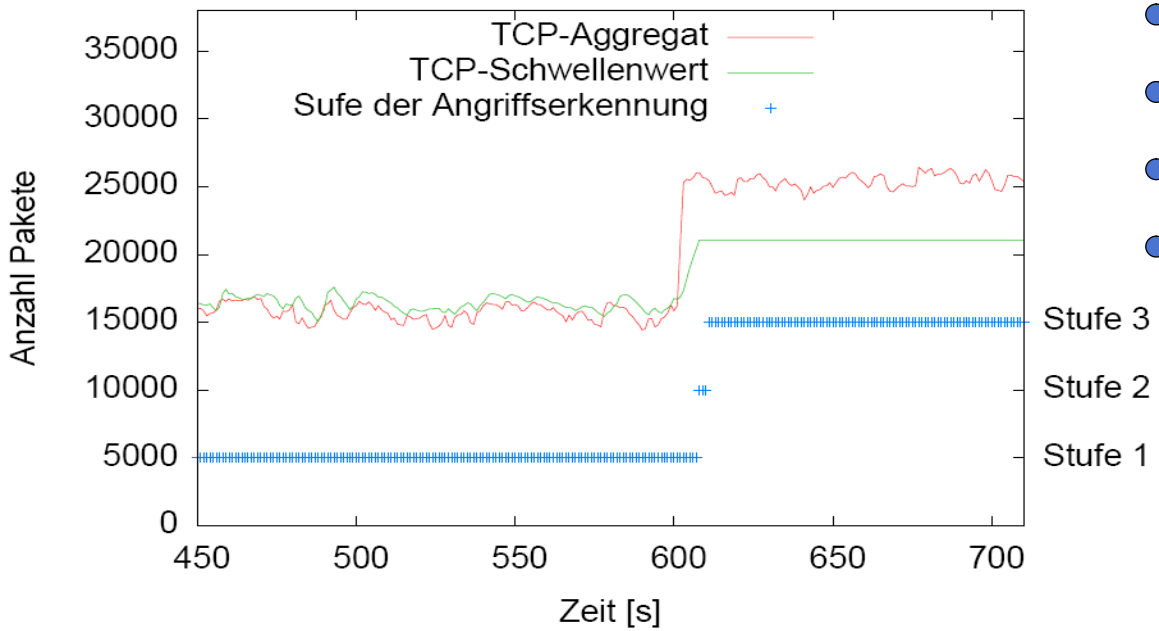
Integrating real world applications into OMNeT++, Institute of Telematics, Universität Karlsruhe (TH), Technischer Bericht, Nr. TM-2008-2, Feb 2008.

Evaluation – Erweiterbarkeit

- Integration einer Angriffserkennung
  - Extraktion der Aufgaben → Zerteilung in Module
  - Erstellung von Nachrichten für Kommunikation
  - Konfiguration gruppiert Module und definiert Granularitätsstufen
- Vorteile
  - Menge an Bausteinen → Wiederverwendbarkeit
  - Verschiedenen Laufzeitumgebungen
    - Z. B. Simulation in OMNeT++
  - Erweiterbarkeit
    - Verteilte Kommunikation leicht integrierbar
    - Module einfach austauschbar
    - Neue Funktionalität durch Module integrierbar

Simulation der Angriffserkennung

- Simulation verteilter Angriffe in OMNeT++
  - Realistische Topologie, selbstähnlicher Verkehr
  - Framework mit Angriffserkennung auf Core-Router
  - 150 DDoS-Zombies auf Endsystemen
  - Opfersystem auf Endsystem

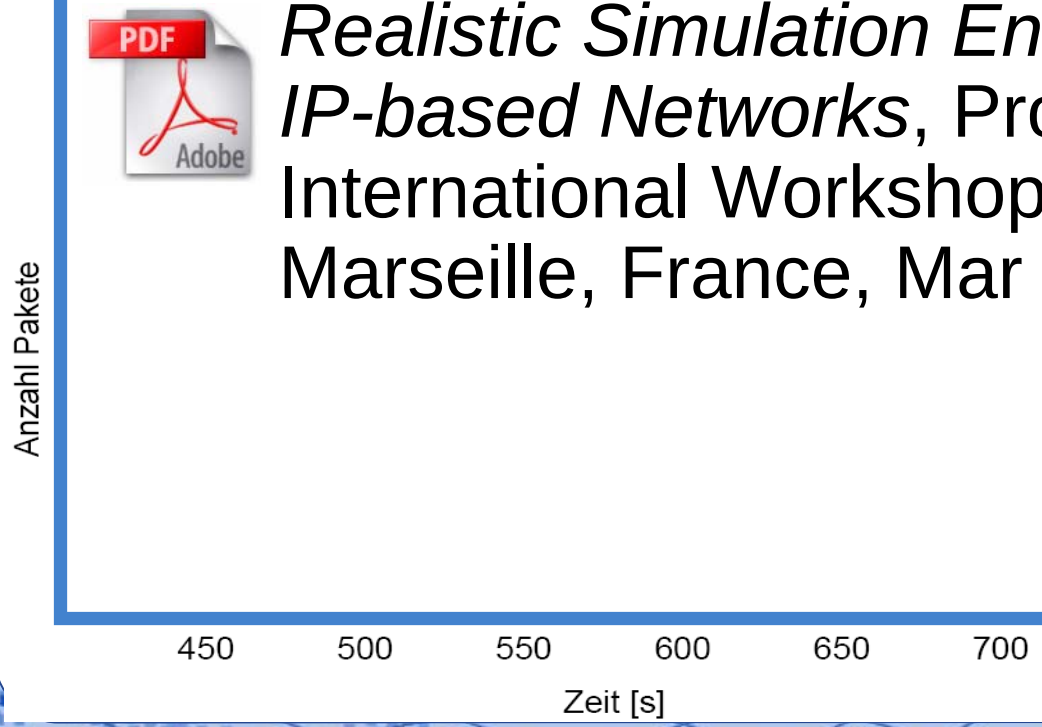


- 3 Core-Router
- 9 Gateway-Router
- 113 Edge-Router
- 3257 Endsysteme

Simulation der Angriffserkennung

- Simulation verteilter Angriffe in OMNeT++

Thomas Gamer und Michael Scharf,
Realistic Simulation Environments for IP-based Networks, Proceedings of 1st International Workshop on OMNeT++, Marseille, France, Mar 2008.



Zusammenfassung und Ausblick

- Framework zur verteilten Angriffserkennung
  - Einfache Integration von neuer Funktionalität
  - Lokale und verteilte Kommunikation
  - Flexibel konfigurierbar und einsetzbar
  - Verschiedene Laufzeitumgebungen
- Ausblick
  - Grafische Oberfläche
  - Weitere Laufzeitumgebungen
  - Neue Methoden zur Angriffserkennung
  - Simulation verteilter Angriffserkennung

Bald als Open Source-Projekt unter
http://www.tm.uka.de/~mayer/distack

Vielen Dank! Fragen?