Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institute of Computer Science
Department of Distributed Systems
Prof. Dr.-Ing. P. Tran-Gia

The Publish-Subscribe Communication Paradigm

(Einsatz und Möglichkeiten von Publish/Subscribe als Ergänzung zu verteilten Datenbanken)

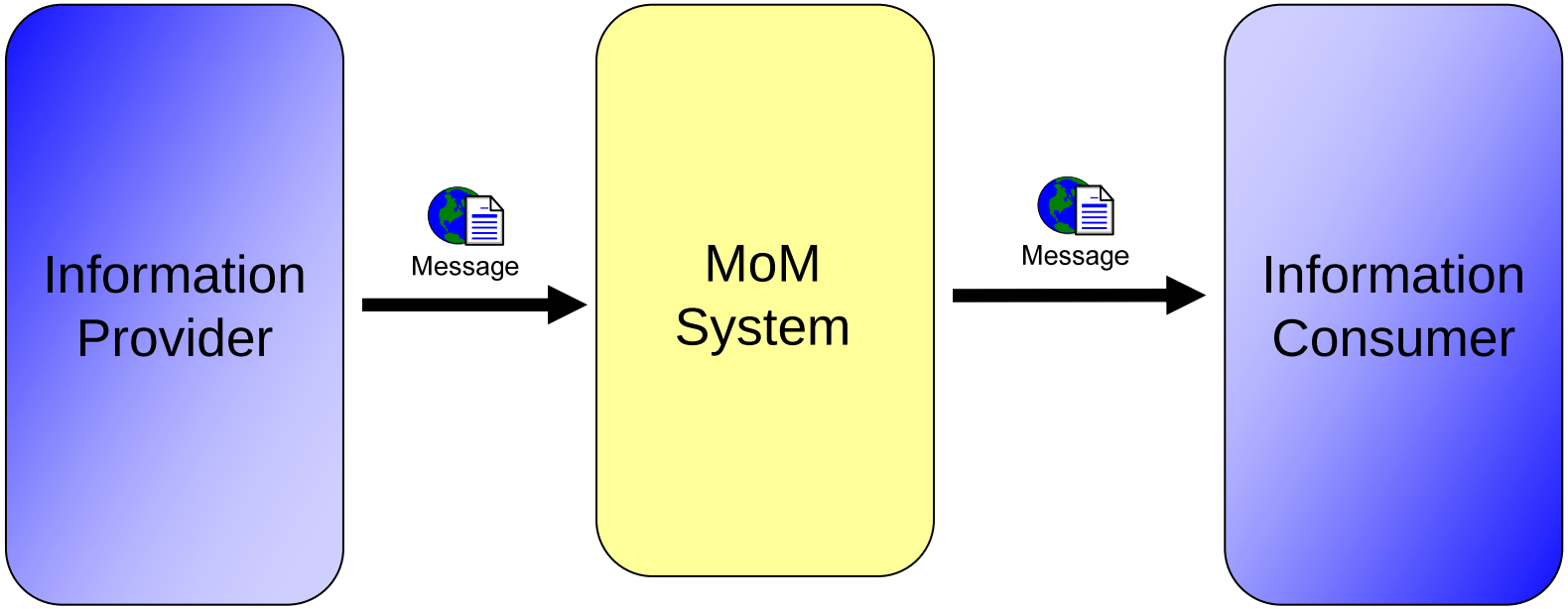
Robert Henjes

www3.informatik.uni-wuerzburg.de

Publish/Subscribe als verteilte Datenbasis

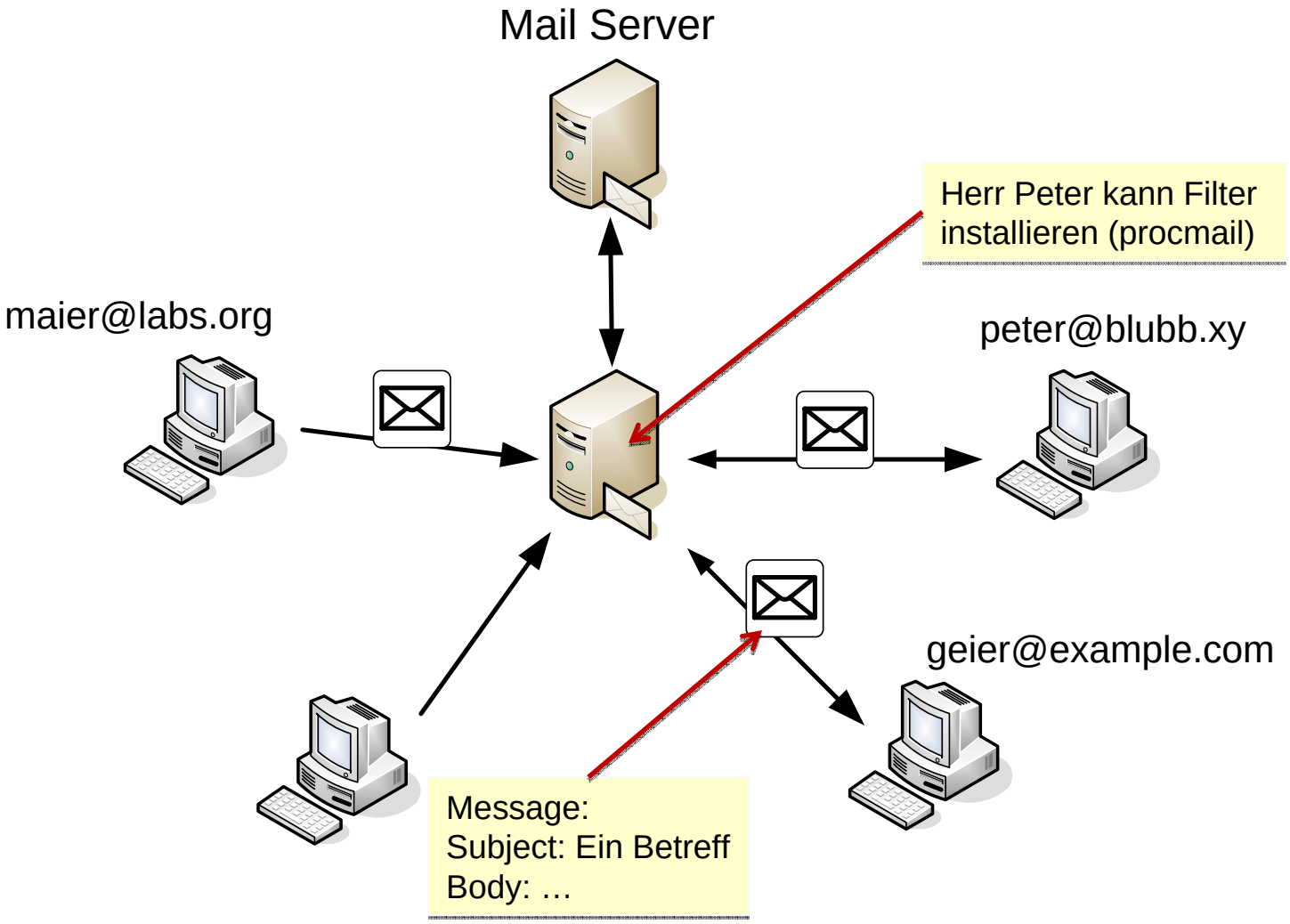
- Das Publish/Subscribe-Prinzip als verteilte Datenbasis
- Das Kommunikationsparadigma und Java Messaging Service
- Bewertung und Analysemethoden zur Charakterisierung der Serverperformance

Message Oriented Middleware: Push-Push Mechanismus

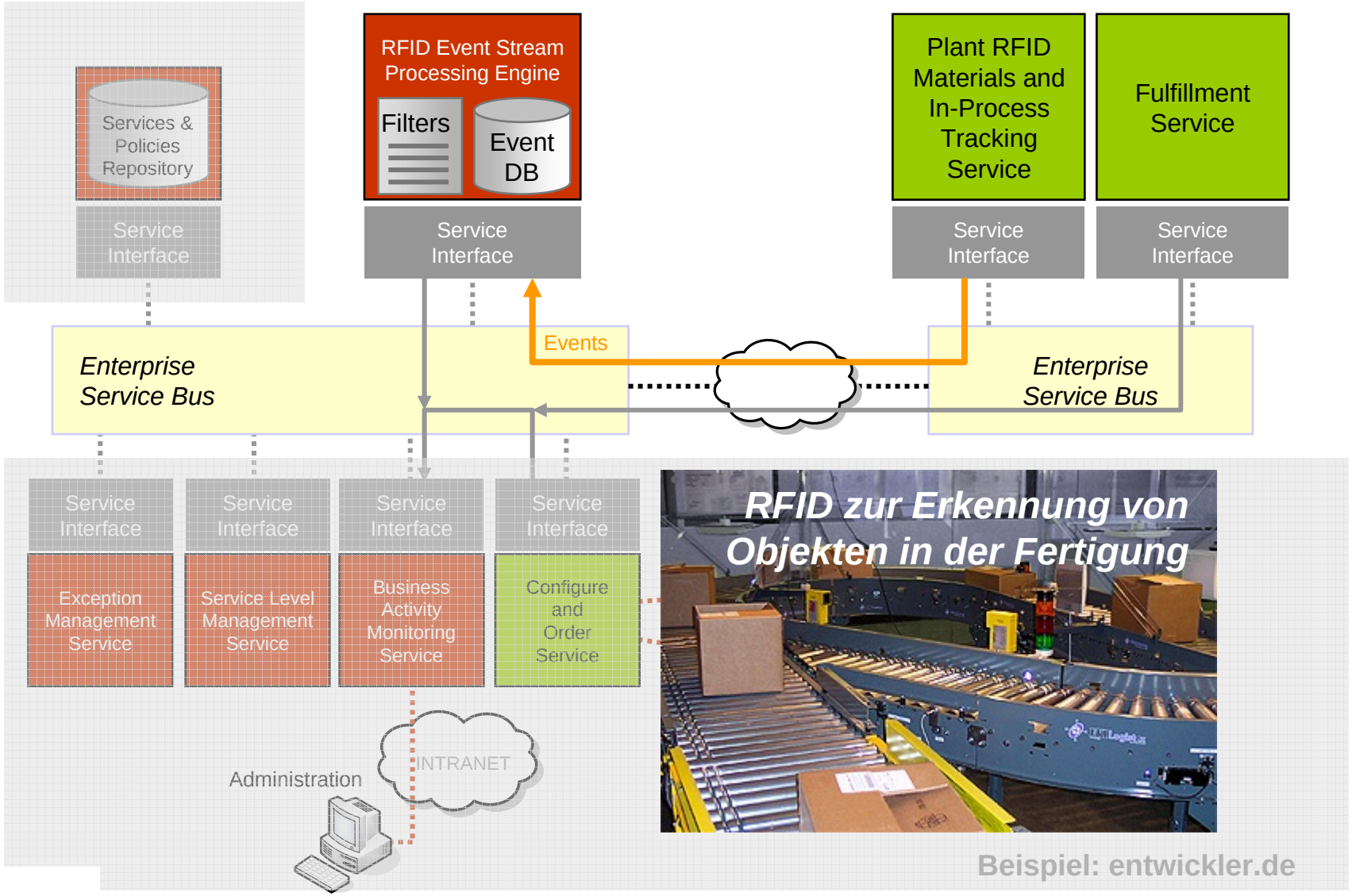


- **Microsoft: Message Queuing Services (MSMQ)**
  - .NET Implementierung eines MoM Systems
- **Java Message Service (JMS)**
  - Die *JMS API* von Sun Microsystems definiert Java Interfaces für MoM Applikationen

Publish/Subscribe – Eine neue Erfindung?

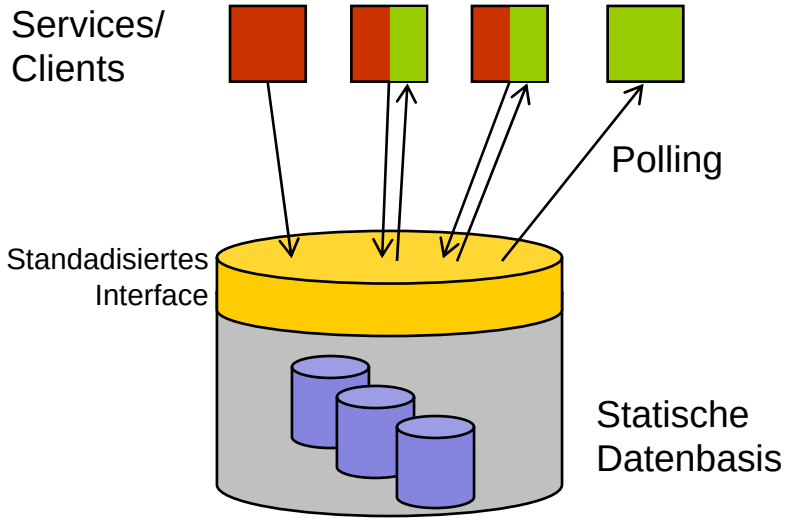


Praxisbeispiel aus dem Fertigungsbereich

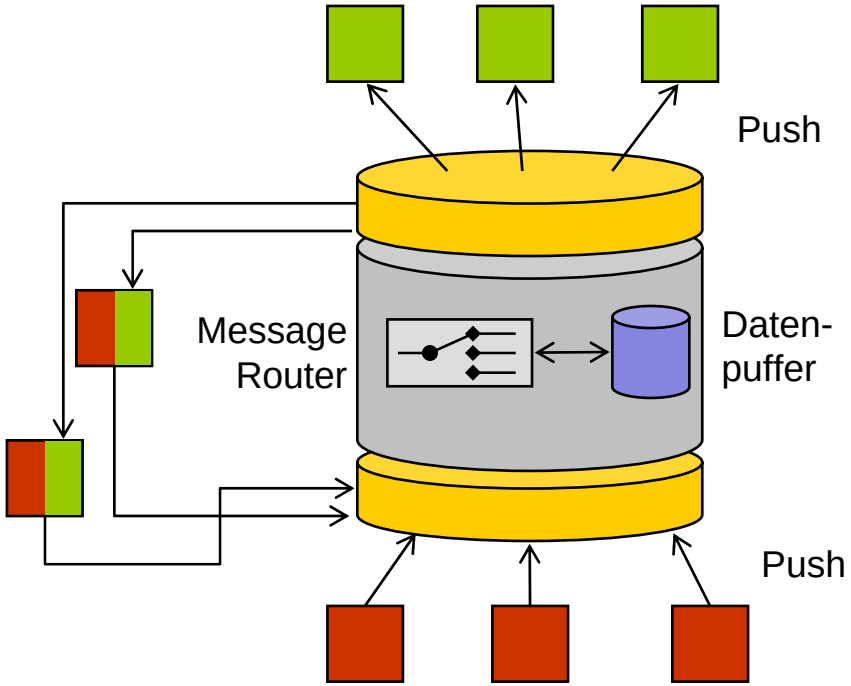


Publish/Subscribe – „Die Live-Datenbank“

Klassische Datenbankanwendung



Publish/Subscribe System



- Klassische Datenbankanwendung
  - Anfragen müssen einzeln im Polling Modus ausgeführt werden
- Einsatz eines Publish/Subscribe Systems
  - Nachrichten werden im Push Betrieb zugestellt

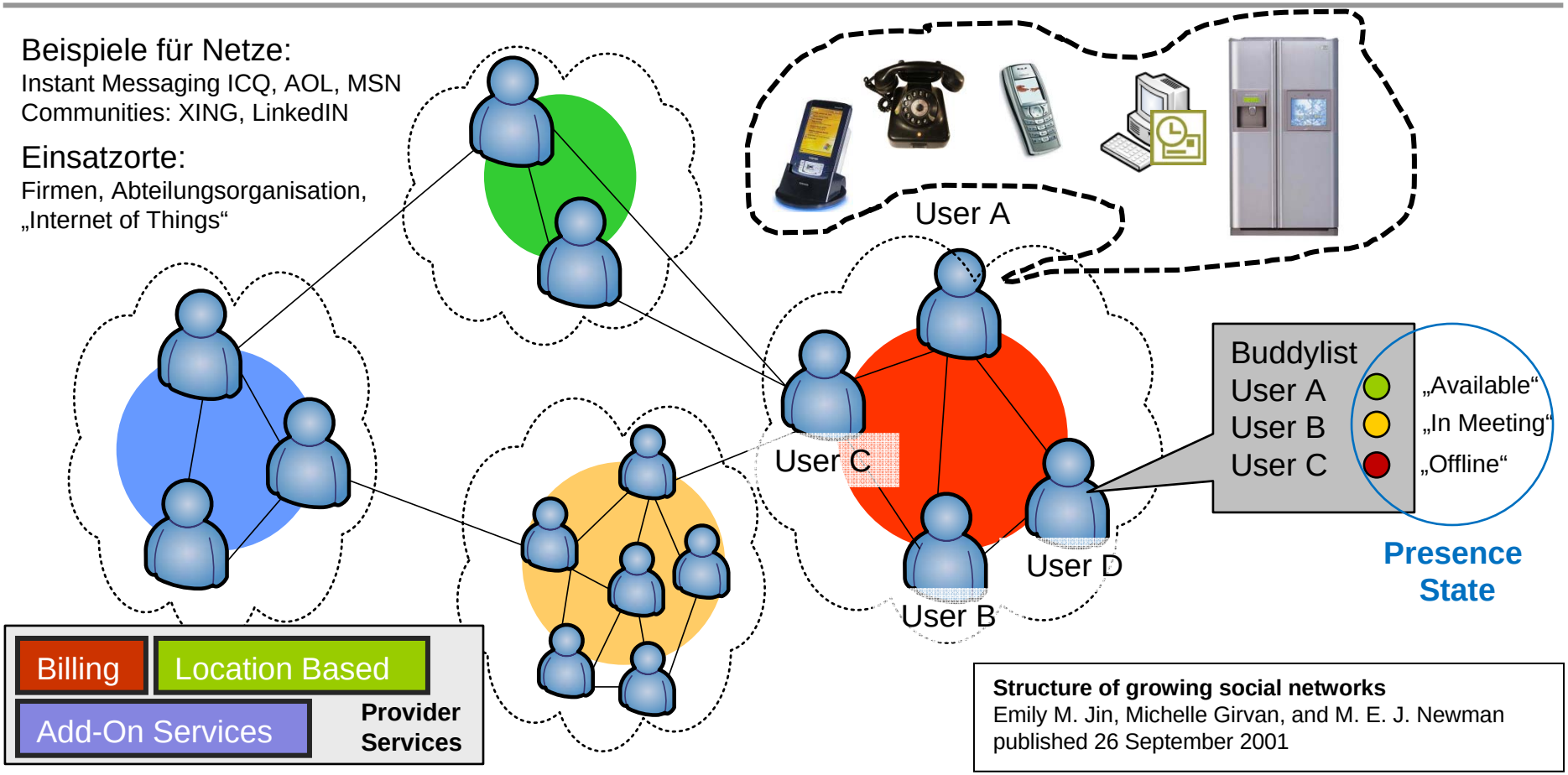
Unterstützte Funktionen in Publish/Subscribe

- **Definition Datenbank:**
  Sammlung von Daten mit Beziehung zueinander
- **Prinzipien beim Datenbank Design**

In Pub/Sub

  - Daten möglichst eindeutig speichern 
  - Plausibilität muss gewährleistet werden 
  - Kontrolle bei gleichzeitigem Zugriff 
  - Zugangskontrolle / Berechtigungen 
- **Protokolle (z.B. bei JMS)**
  - Kommunikation basiert meist auf XML
  - Subscription Sprache basiert auf SQL
- **Betriebsmodalitäten (z.B. bei JMS)**
  - Persistent / Non-Persistent (Garantie der Zustellung)
  - Durable / Non-Durable (Zwischenspeicherung)
  - Transacted (Rollback - Support)

Anwendungsfall – Presence



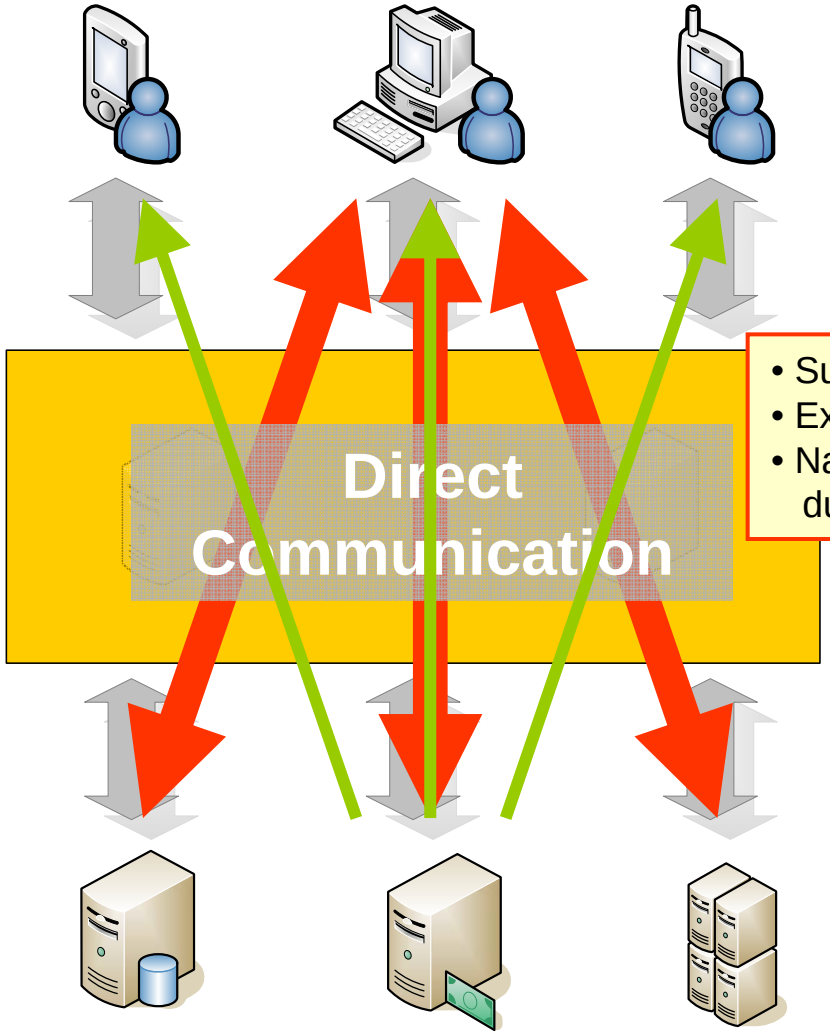
- Zukünftig werden Informationsveränderungen an verschiedenen Komponenten benötigt
- Speicherung des Zustandes an einer Stelle muss vielen Anfragen standhalten

Kommunikationsstrukturen auf Applikationsebene

Consumer

Kommunikations-infrastruktur

Services



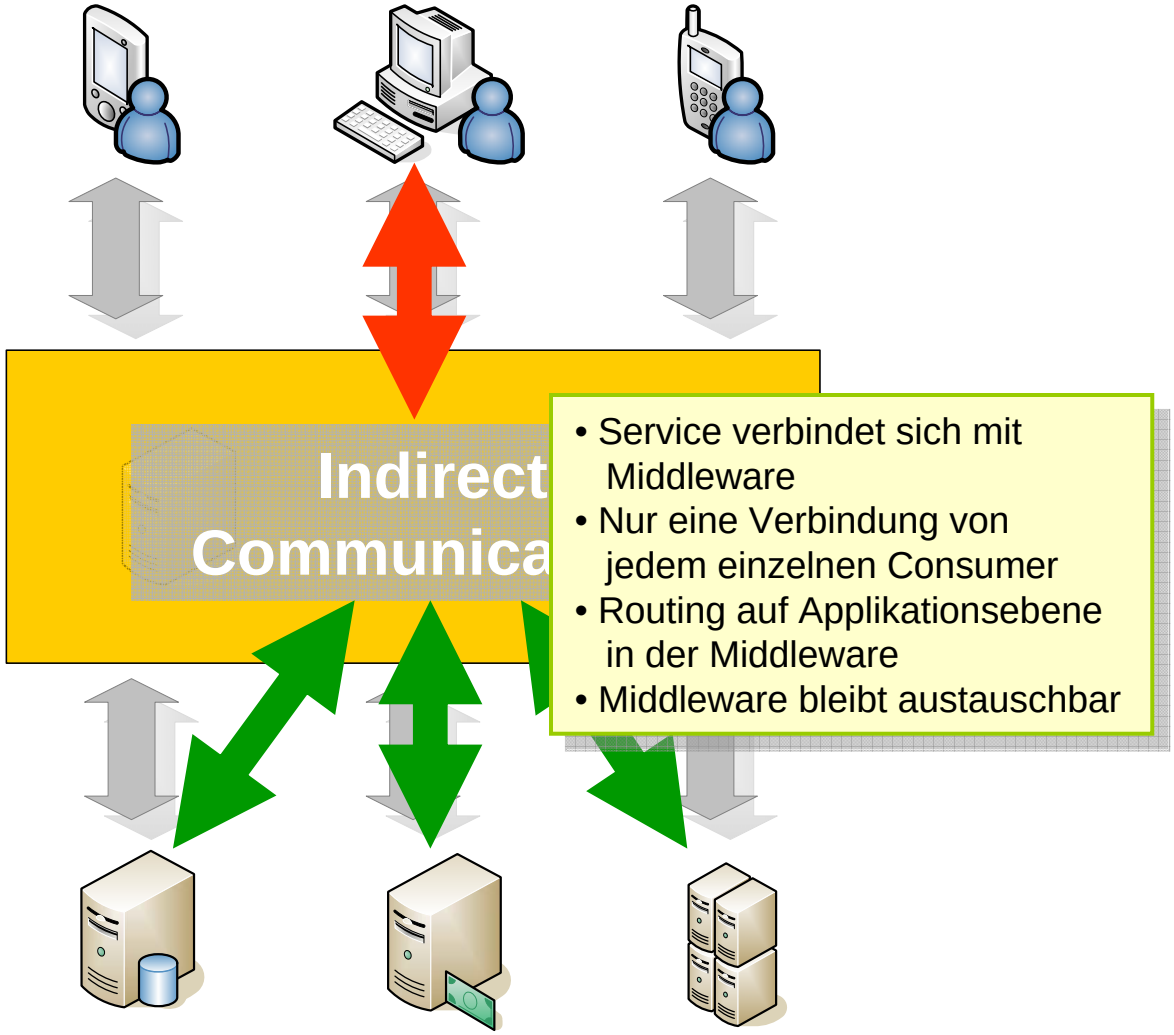
- Suche nach Service
- Exklusive Verbindungen
- Nachrichtenverteilung durch jeden Service

Kommunikationsstrukturen auf Applikationsebene

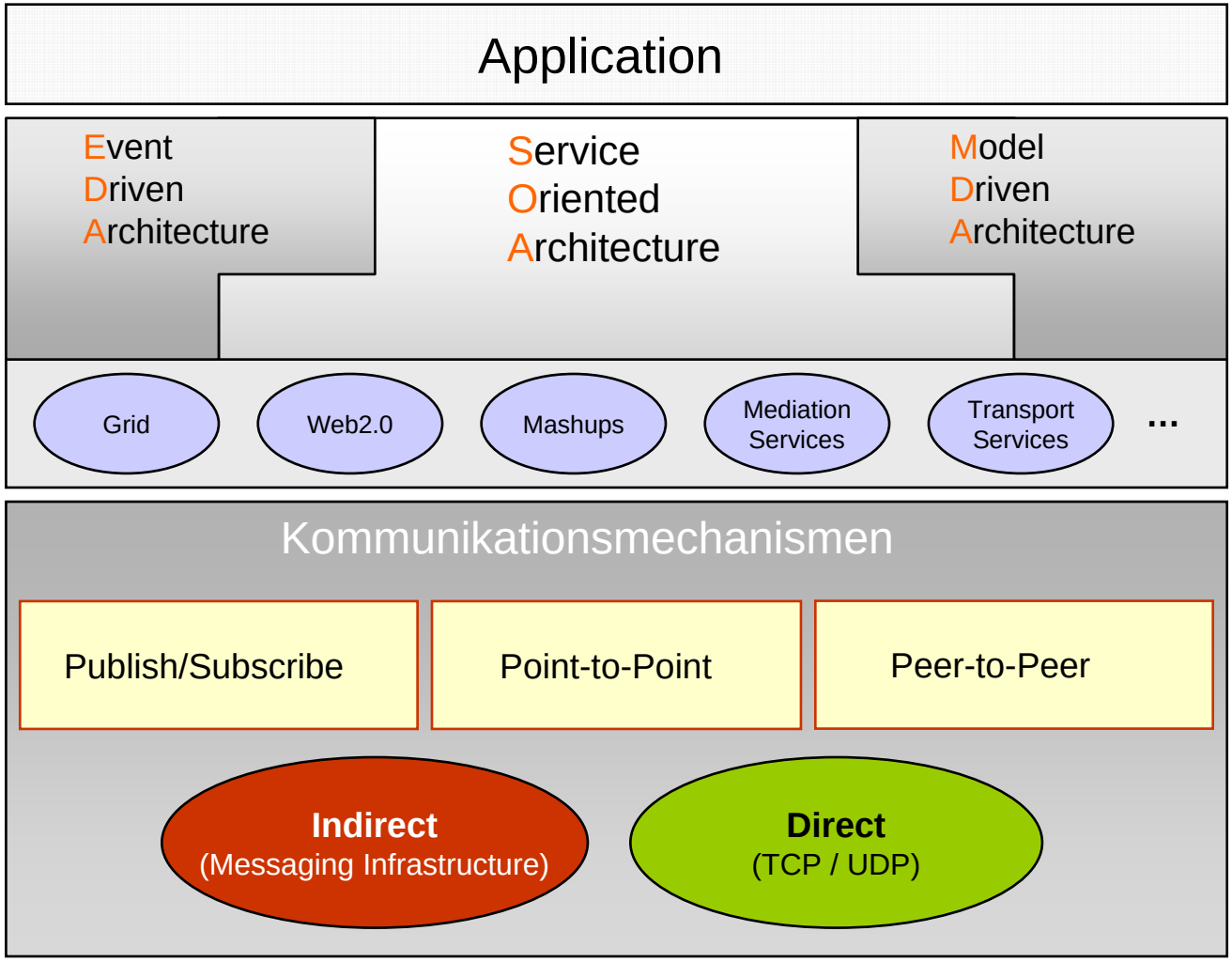
Consumer

Kommunikations-infrastruktur

Services



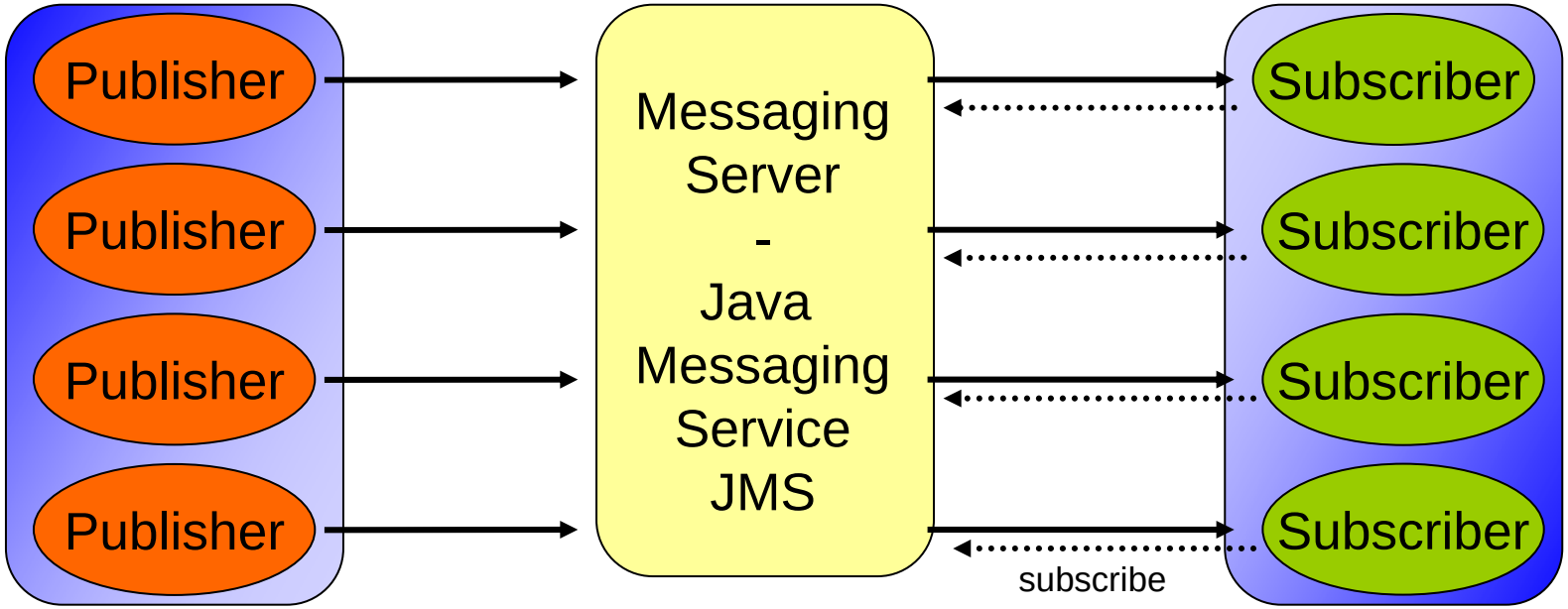
Einordnung verschiedener Architekturen und Mechanismen



Publish/Subscribe als verteilte Datenbasis

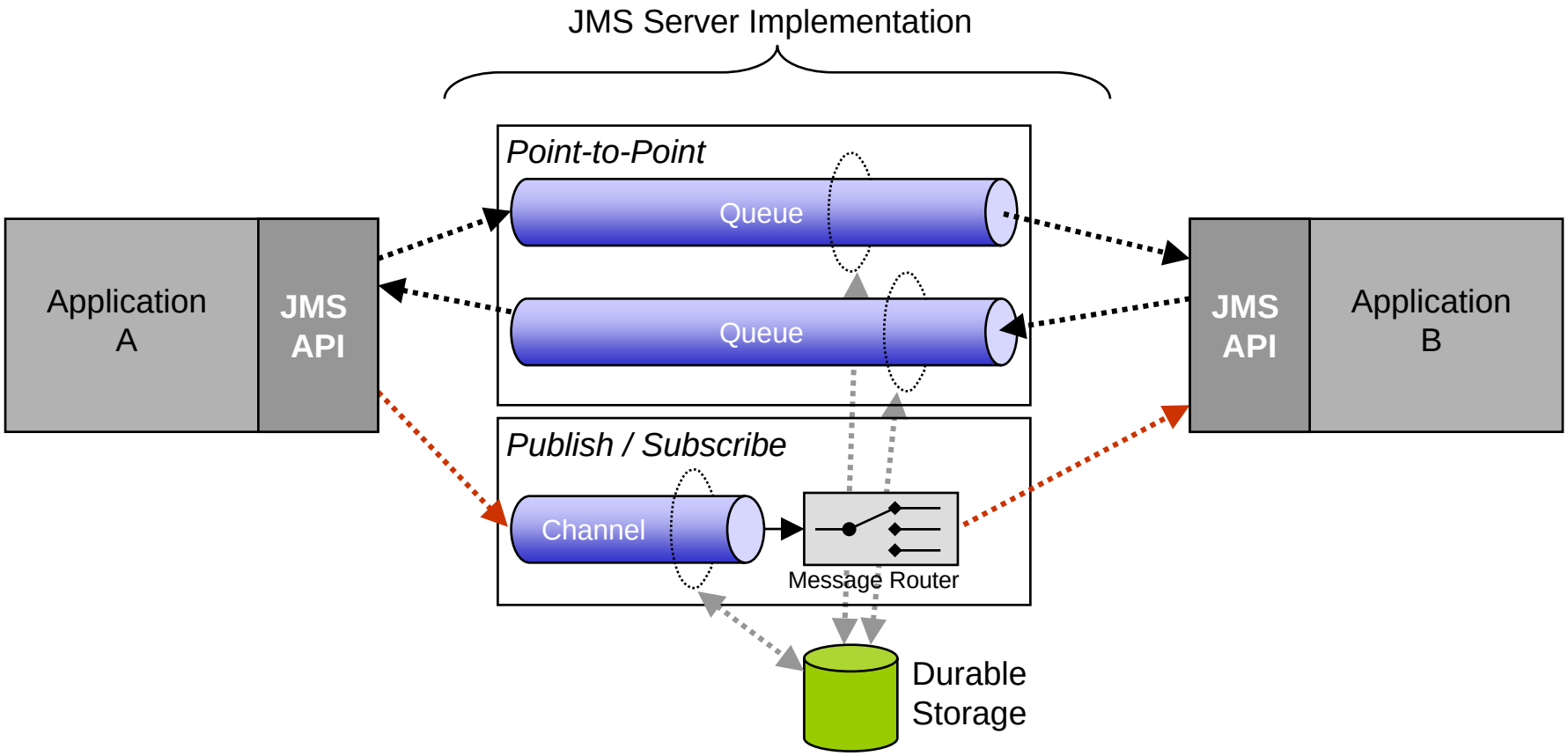
- Das Publish/Subscribe-Prinzip als Datenbankunterstützung
- Das Kommunikationsparadigma und Java Messaging Service
- Bewertung und Analysemethoden zur Charakterisierung der Serverperformance

Das Publish / Subscribe Prinzip formell



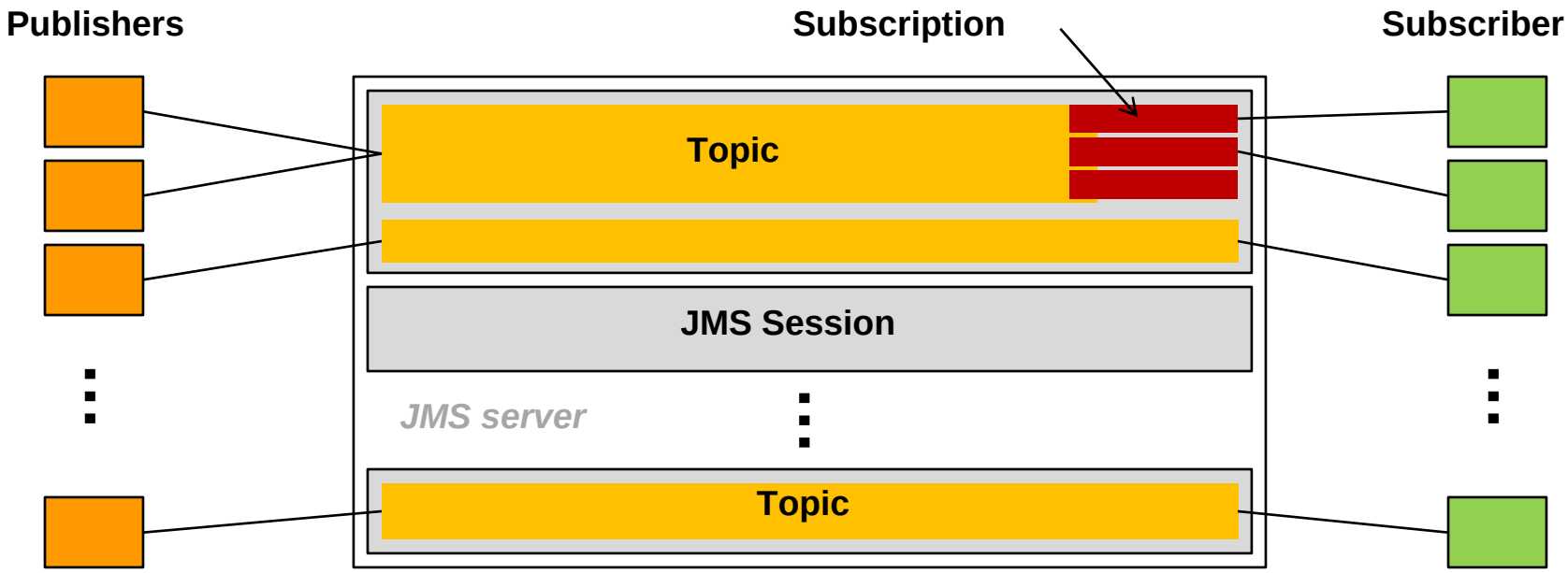
- Entkopplung von Informationsprovidern und Consumern in
  - Raum (kein direktes Wissen über die Kommunikationspartner)
  - Zeit (Teilnehmer müssen nicht gleichzeitig erreichbar sein)
  - Synchronität (Die Kommunikationspartner blockieren sich nicht gegenseitig bei der Kommunikation)

Strukturierung des Java Messaging Service (JMS)



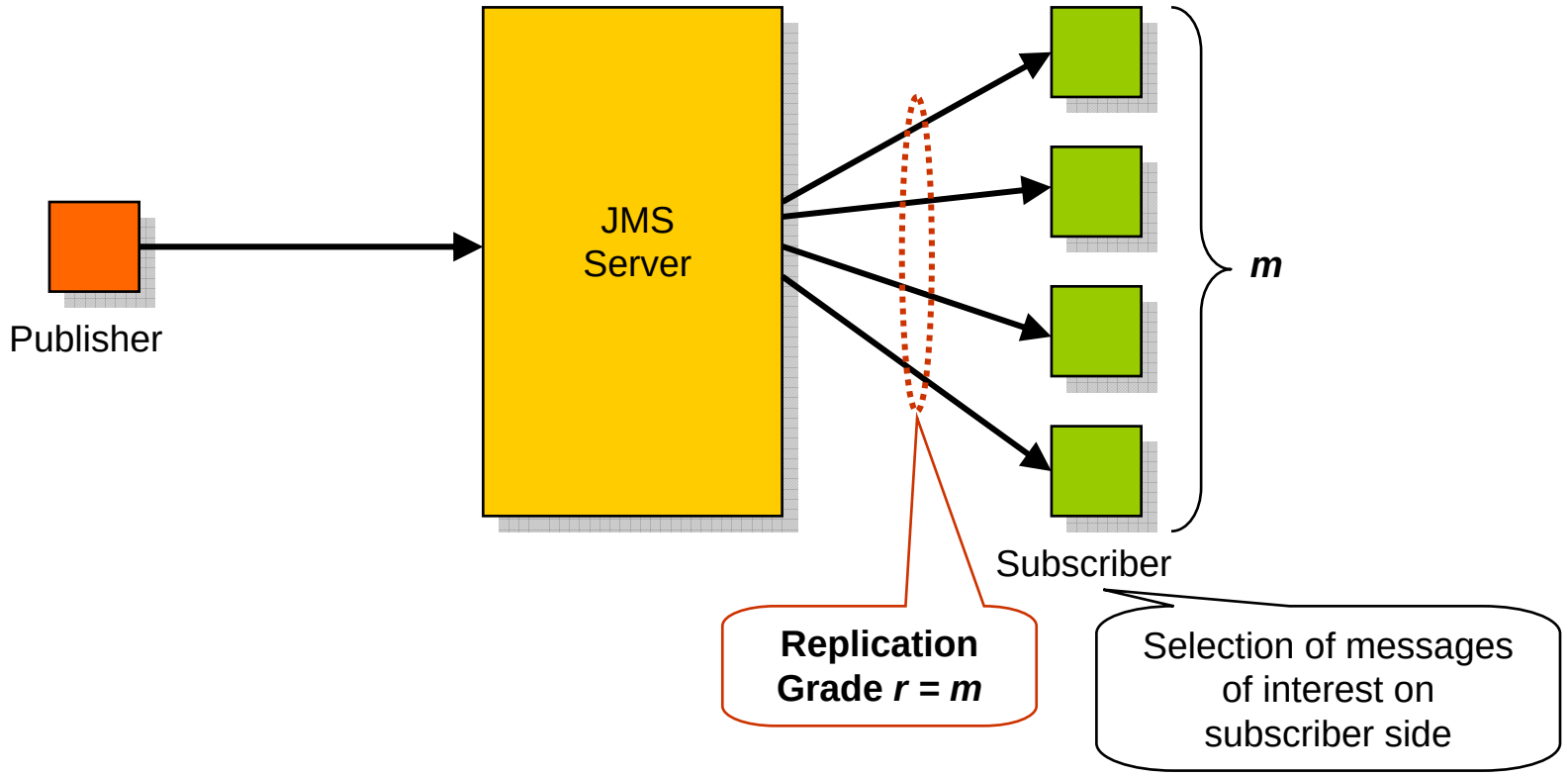
- Zwei verschiedene Kommunikationsmuster
  - Point-to-Point: Realisiert durch unidirektionale Queues
  - Publish/Subscribe: Gerichteter Kommunikationskanal

Options for Grouping Subscriptions in JMS



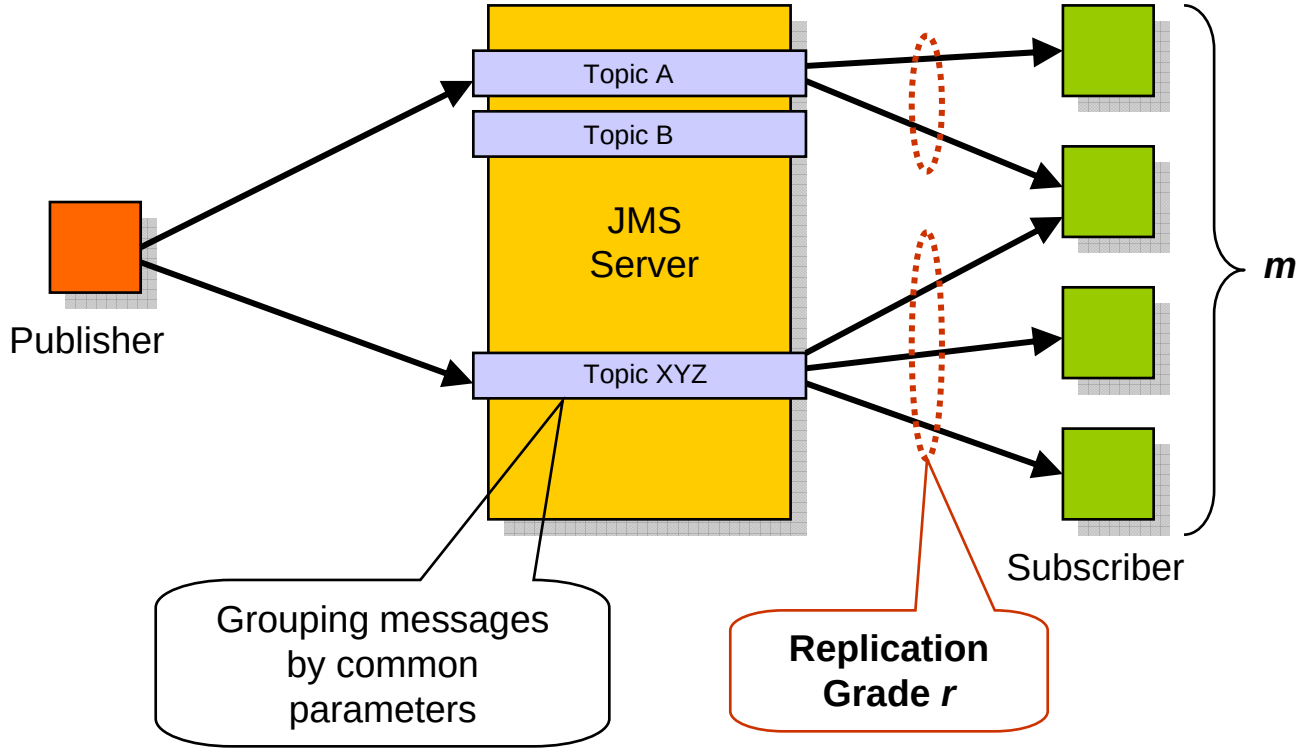
- Options of grouping interests in a JMS system:
  - TCP/UDP connection on network layer
  - JMS session
  - *JMS topic*
  - JMS subscription
  - *Logical operators within a subscription*

Nachrichtenrouting – Message Selection (Subscriber seitig)



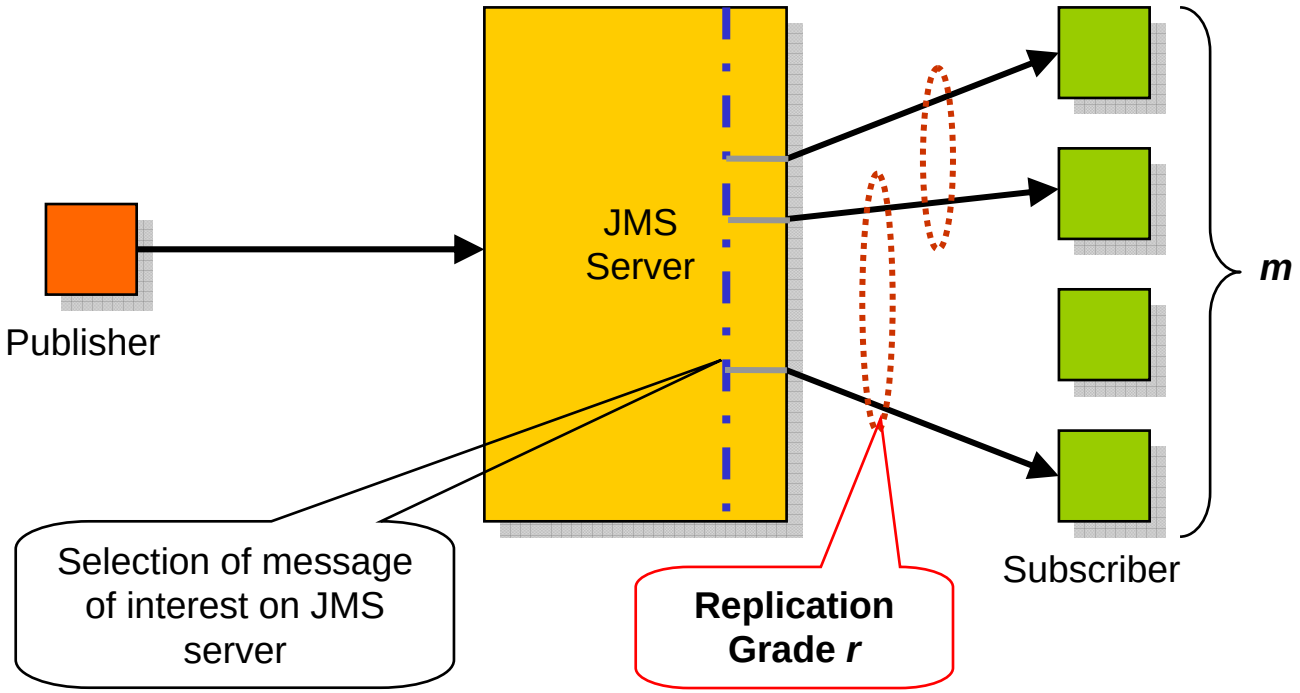
- Subscriber empfangen alle gesendeten Nachrichten
- Replikationsgrad entspricht der Anzahl der verbunden Subscriber

Nachrichtenrouting – Message Selection (Topics)



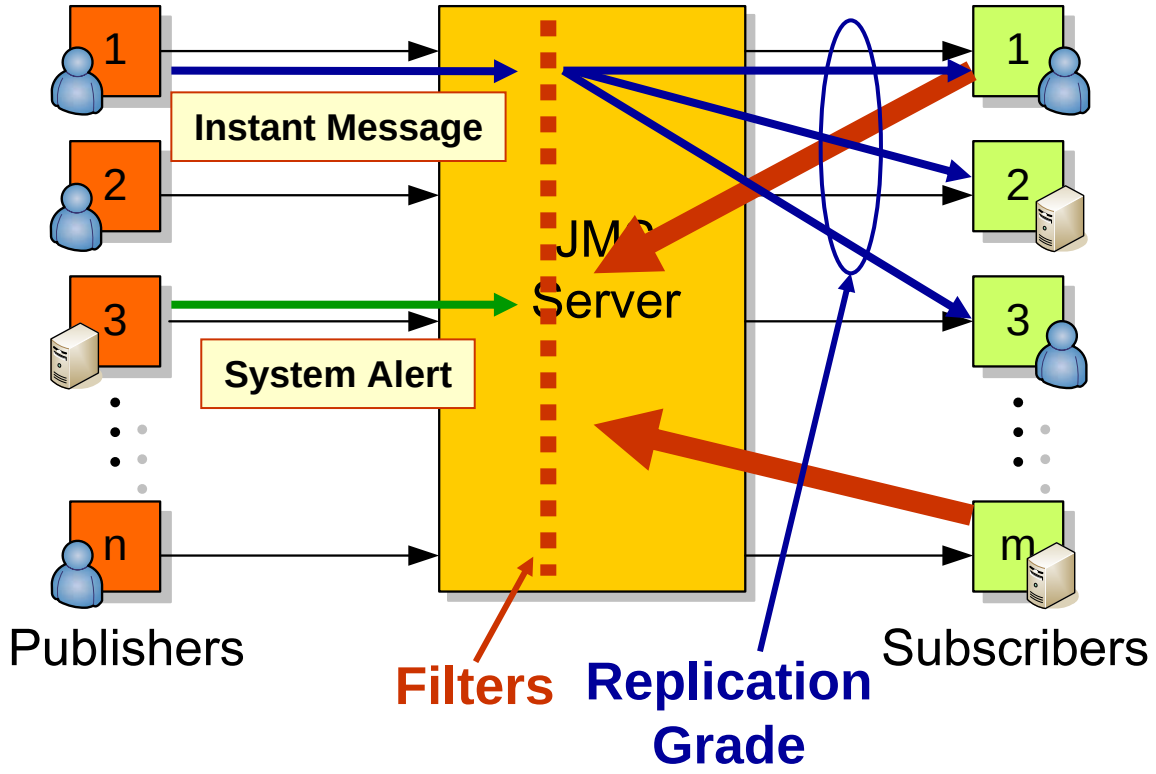
- Subscriber empfangen nur Nachrichten die an ein bestimmtes Topic gesendet wurden
- Replicationsgrad entspricht der Anzahl verbundener Subscriber pro Topic

Nachrichtenrouting – Message Selection (Filtering)



- Subscriber empfangen nur Nachrichten für die sie einen Filter auf dem Server gesetzt haben
- Replicationsgrad entspricht der Anzahl von Subscribern, die sich für die entsprechende Nachricht interessieren

Die verschiedenen Message Routing Optionen

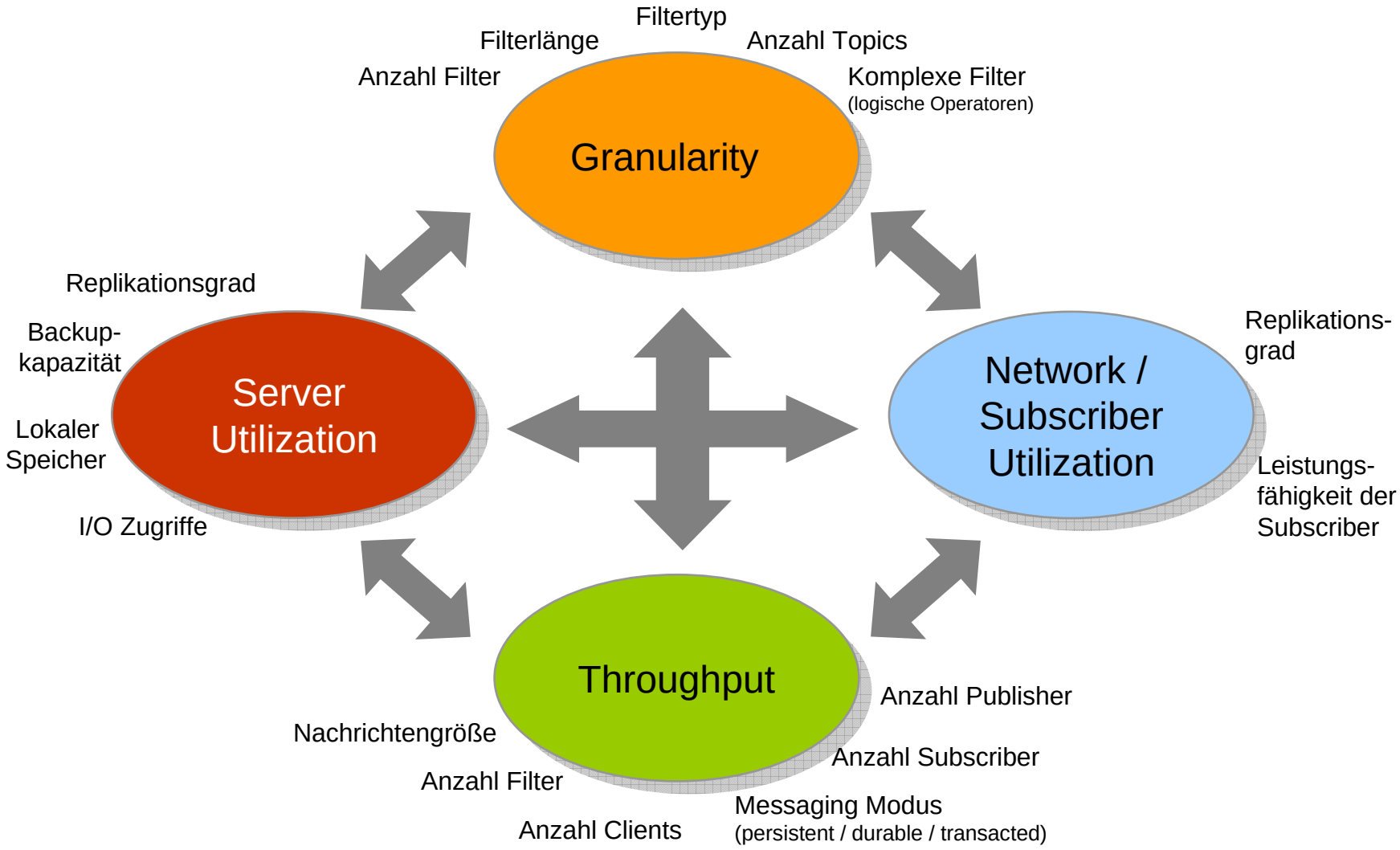


- Welche Parameter beeinflussen die JMS Server Performance?
- Wie viele Nachrichten kann ein JMS Server pro Sekunde verarbeiten?
- Inwieweit hängt die Kapazität vom Anwendungsfall ab?

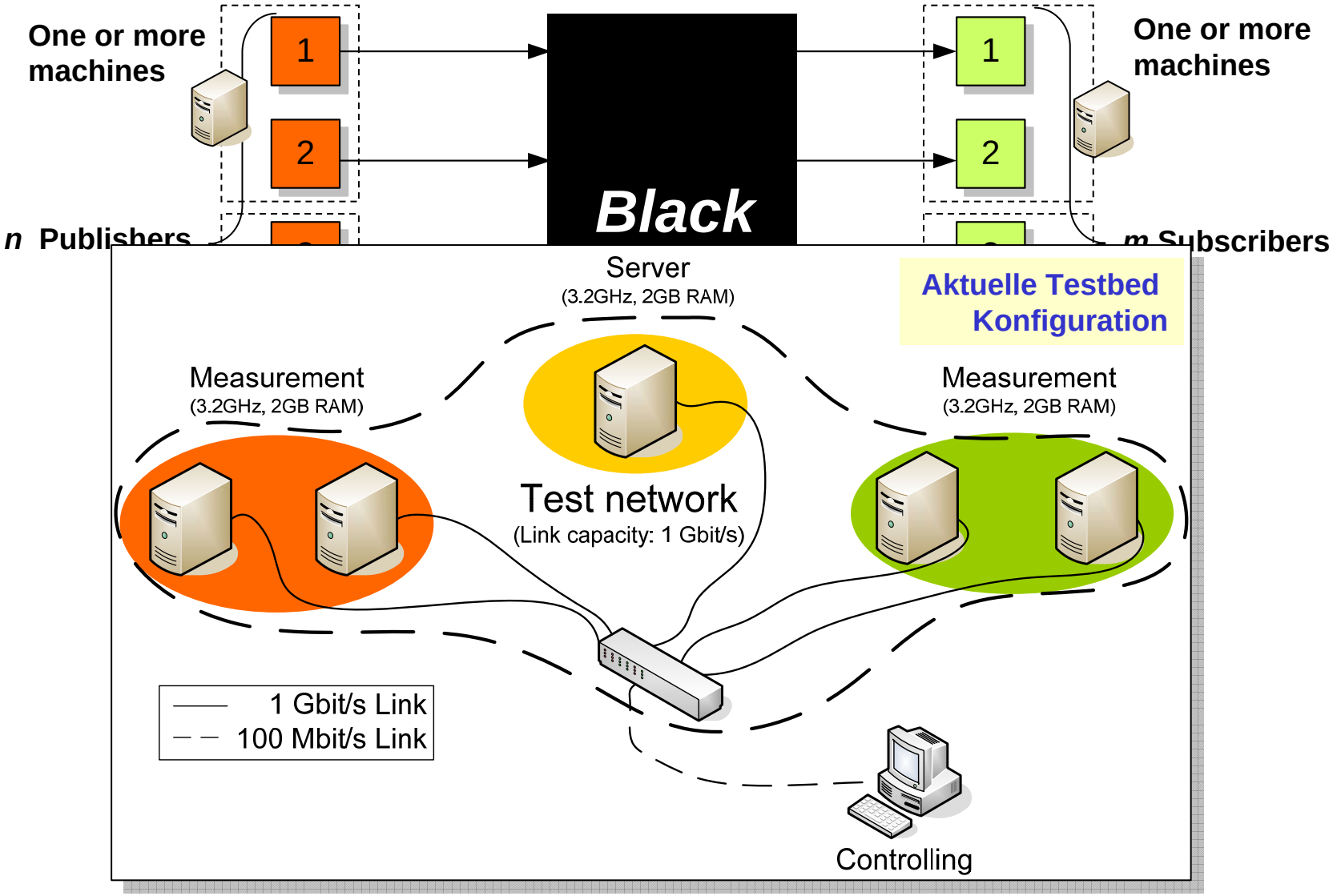
Publish/Subscribe als verteilte Datenbasis

- Das Publish/Subscribe-Prinzip als Datenbankunterstützung
- Das Kommunikationsparadigma und Java Messaging Service
- Bewertung und Analysemethoden zur Charakterisierung der Serverperformance

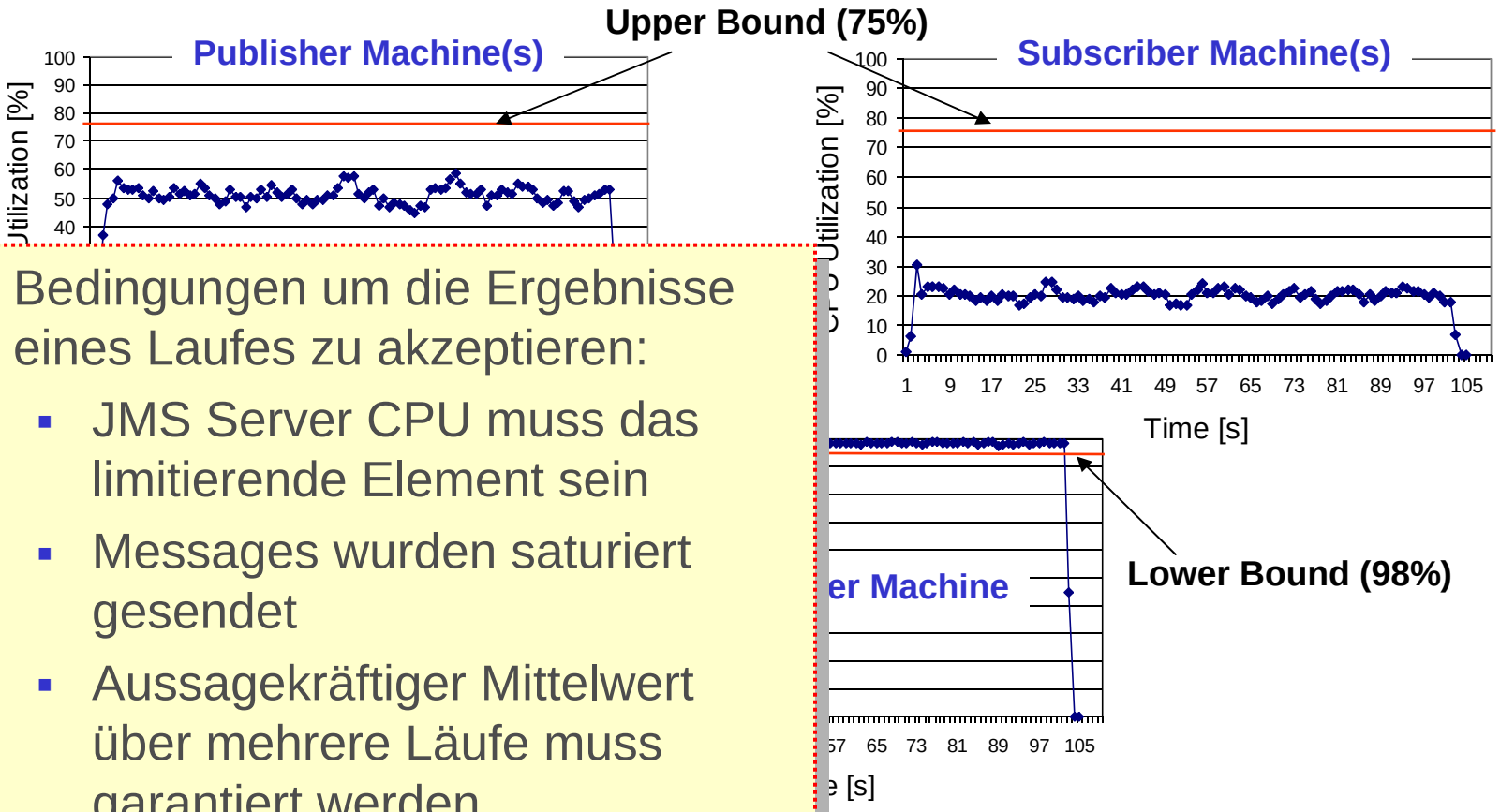
Einflussfaktoren und Tradeoffs für die Leistung



Test Umgebung – Soft- und Hardware

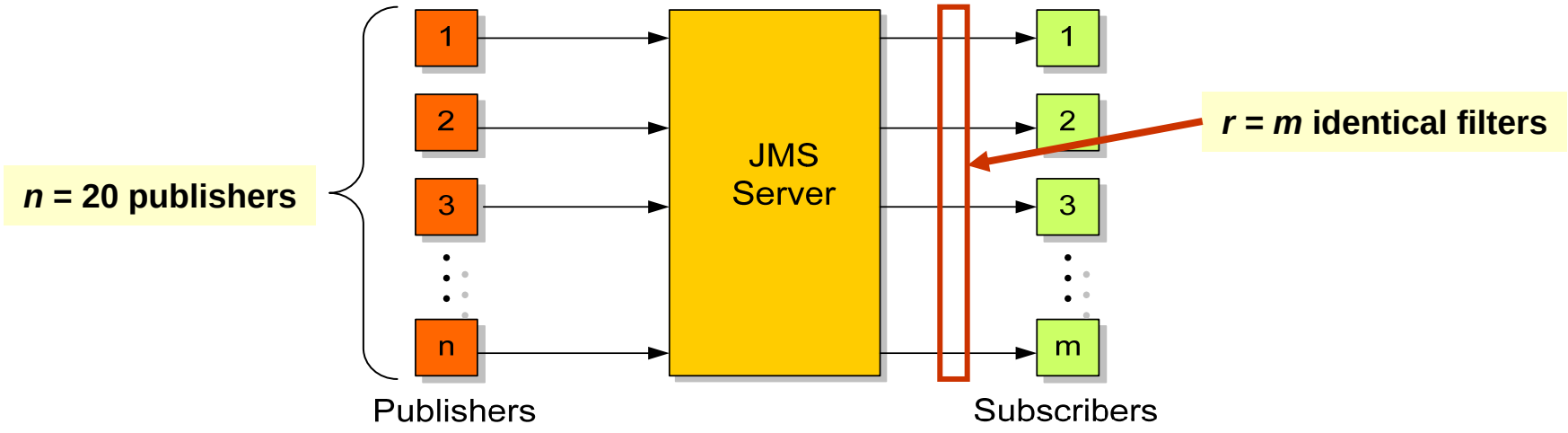


Methodik für die Messungen



- Bedingungen um die Ergebnisse eines Laufes zu akzeptieren:
  - JMS Server CPU muss das limitierende Element sein
  - Messages wurden saturiert gesendet
  - Aussagekräftiger Mittelwert über mehrere Läufe muss garantiert werden
- Die CPU Last auf Publisher und Subscriber Rechnern darf 75% überschreiten.
- Die Idle Zeit der Server CPU darf 2% nicht übersteigen.

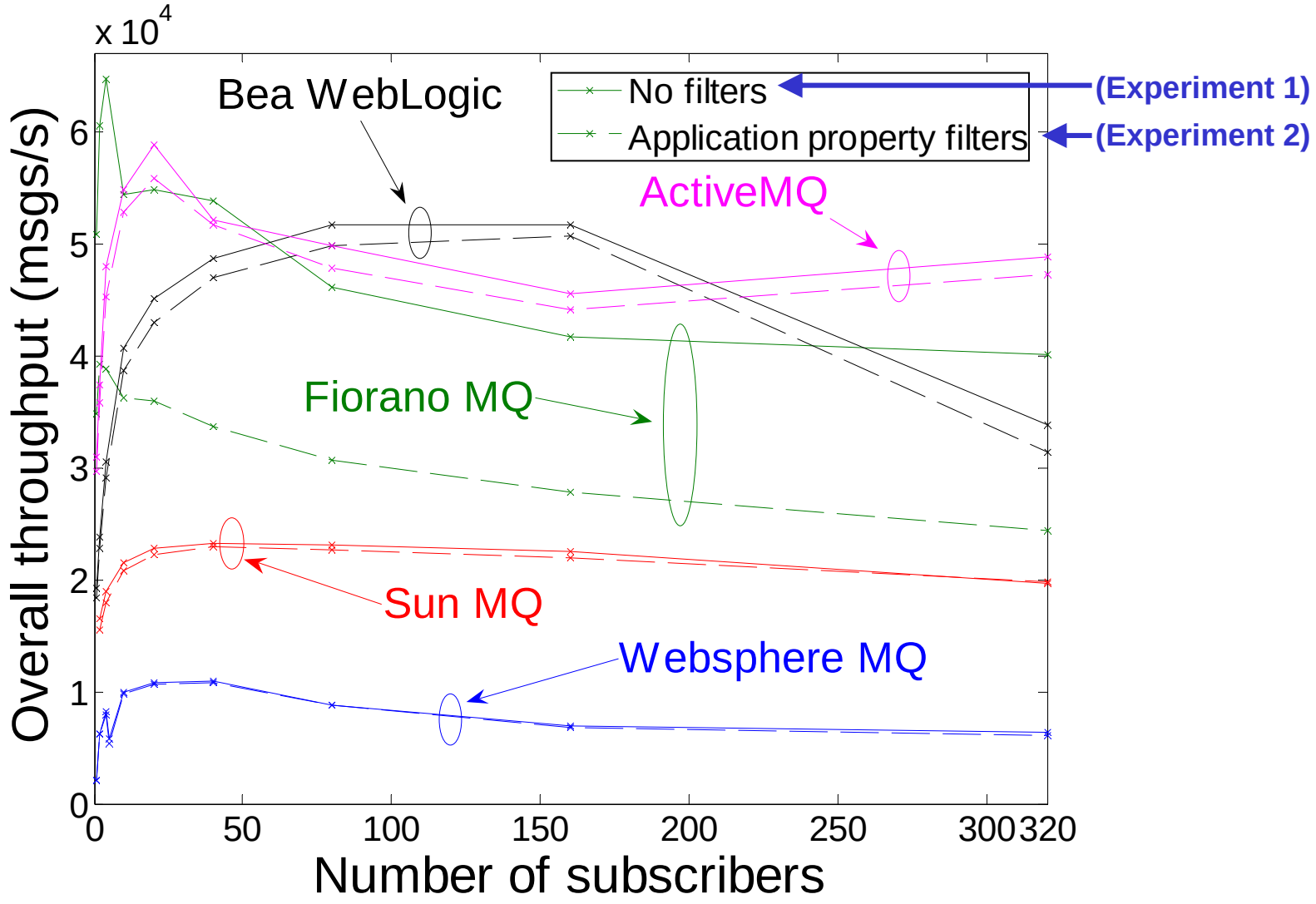
Versuchsaufbau für die Messung einfacher Filter



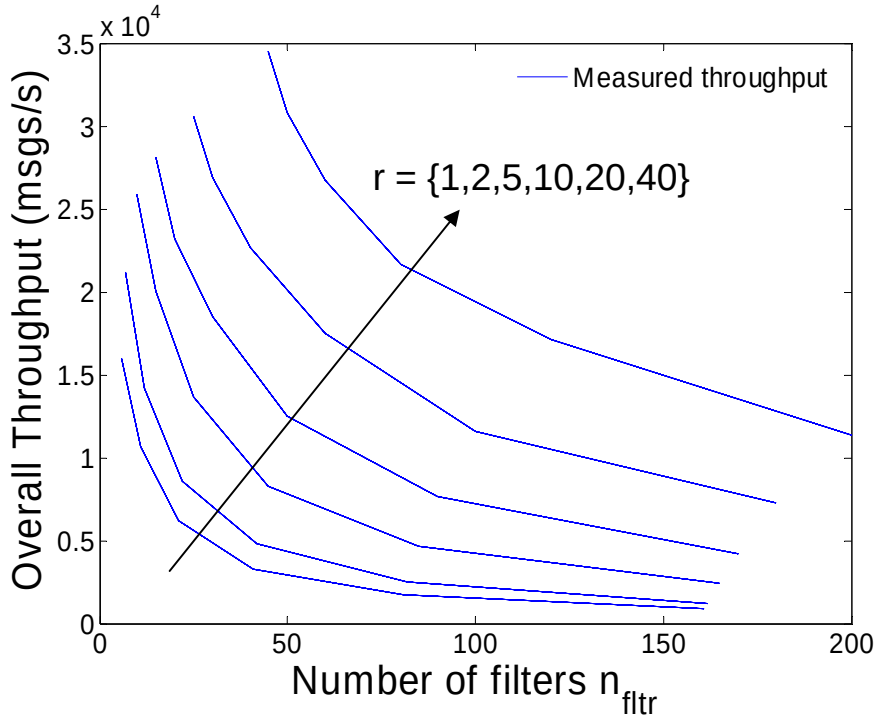
- 4 Rechner insgesamt: 1 Publisher, 1 JMS server, 2 Subscriber
- *n* = 20 Publisher senden Nachrichten mit AppProp ID *#0*

- *Keine Filter auf dem JMS server installiert* (Experiment 1)
  ⇒ Replikationsgrad *r* = *m*
- *Application header properties (AppProp)* (Experiment 2)
  - Jeder Subscriber filter nach der ID *#0*
  - *m* Subscriber empfangen alle gesendeten Nachrichten
    ⇒ Replikationgrad *r* = *m*

Messergebnisse für keine und einfache Filter im Vergleich

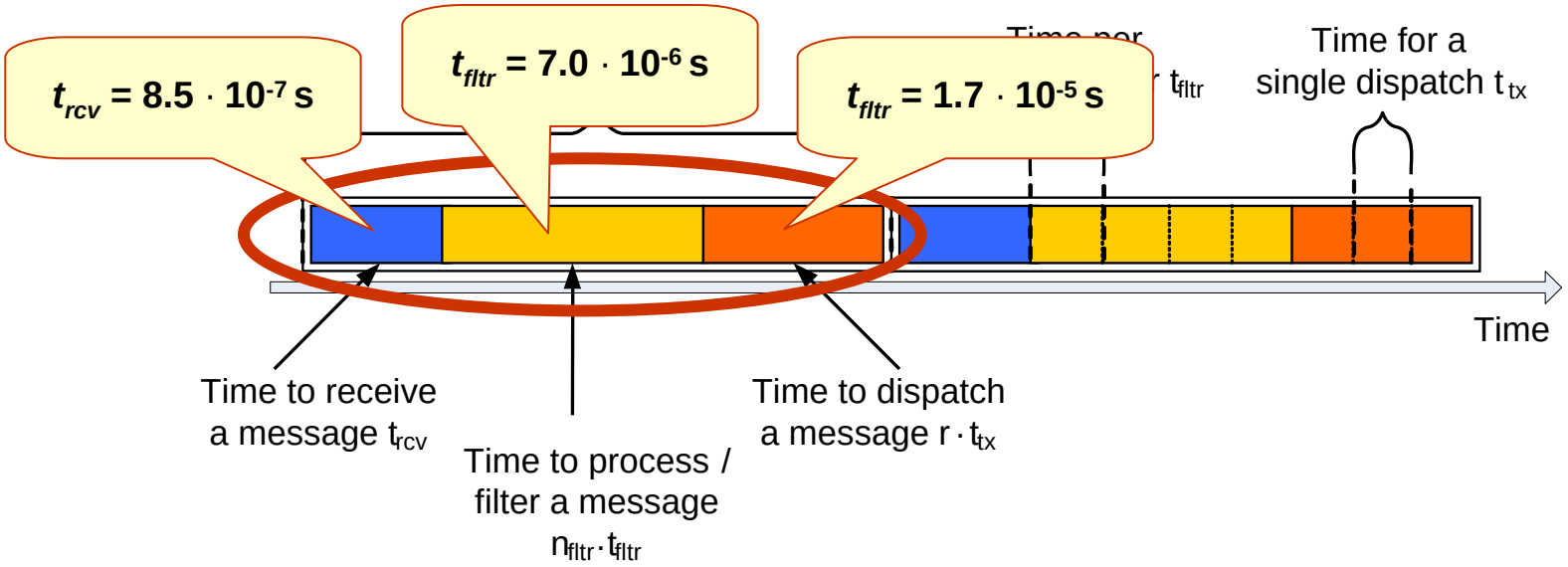


Maximaler Nachrichtendurchsatz für den FioranoMQ



- Steigende Anzahl von Filter verringert den Nachrichtendurchsatz (trivial)
- Steigender Replikationsgrad von Nachrichten
  - verringert der Durchsatz auf Publisherseite
  - erhöht den gesamten Nachrichtendurchsatz

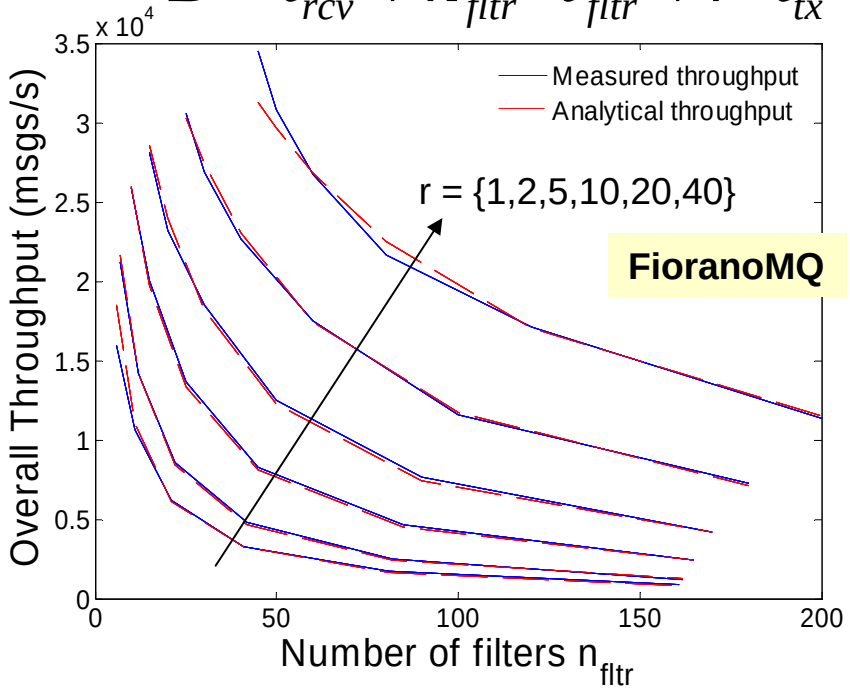
Einfaches analytisches Modell für den Fiorano MQ



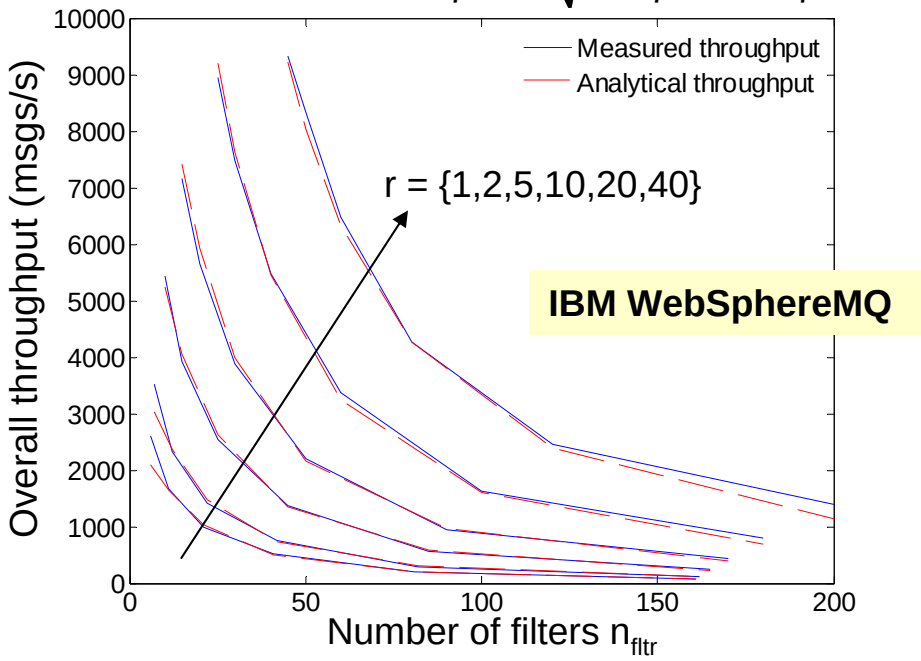
- Jede Nachricht braucht CPU Zeit
  - zum Empfang (t_{rcv})
  - zur Verarbeitung (pro Filter!) ($n_{fltr} \cdot t_{fltr}$)
  - zum Zustellung (t_{tx})
- Nachrichtenverarbeitungszeit: $B = t_{rcv} + n_{fltr} \cdot t_{fltr} + r \cdot t_{tx}$
- Analytisch berechneter Durchsatz: $\frac{1}{B}$

Validierung des analytischen Modells

$$B = t_{rcv} + n_{fltr} \cdot t_{fltr} + r \cdot t_{tx}$$



$$B = t_{rcv} + n_{fltr} \cdot \sqrt{(n_{fltr})} \cdot t_{fltr}$$



- Gute Übereinstimmung des gemessenen mit dem modellierten Durchsatz
- Deutlich sichtbarer Unterschied zwischen den beiden JMS Servern FioranoMQ und IBM WebsphereMQ

Zusammenfassung

- Publish/Subscribe als dynamische Datenbasis
  - Günstige „Schreib“-Operationen für häufig benötigte Daten
  - Datenhaltung beim Client
  - Gut einsetzbar in Event-orientierten Umgebungen

- Point-to-Point
  - Unterstützt Request/Reply Pattern
  - Gut kombinierbar mit Publish/Subscribe

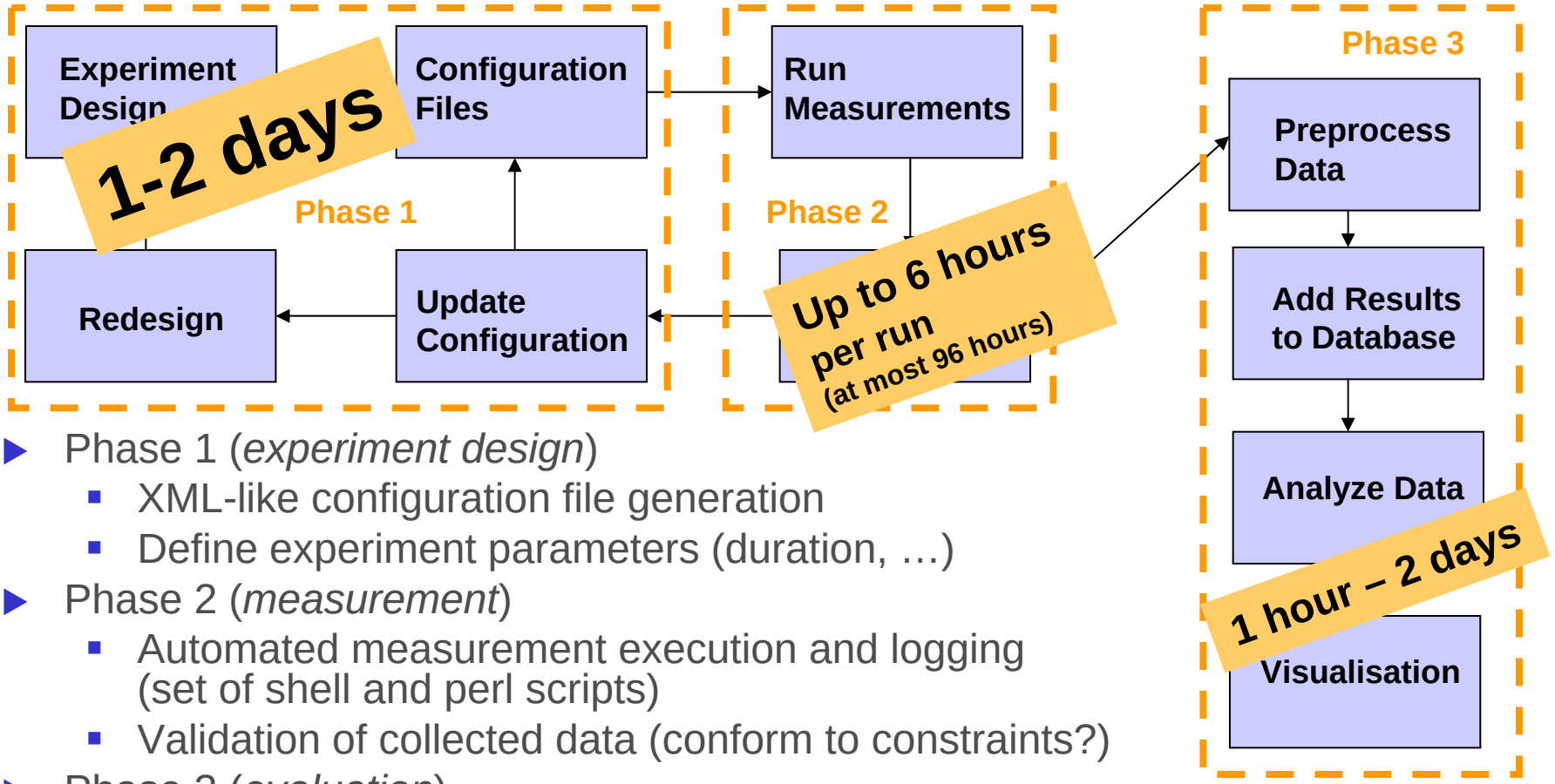
- Publish/Subscribe als Kommunikationsmuster
  - Zentraler, austauschbarer Punkt
  - Unterstützung entkoppelter Kommunkationspartner
  - Flexible Nachrichtenzustellung für jede Art von Service
  - Inhaltsbasiertes Nachrichtenrouting

Literatur

- *Hub Vandervoort,*
  Das Nervensystem des Geschäfts, April 2006
  (http://entwickler-magazin.de/zonen/portale/psecom,id,169,online,817,p,0.html)
- *Gregor Hohpe, Bobby Wolf, et. al.,*
  Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, August 2006, Addison-Wesley
- *Dirk Krafzig, Karl Banke, Dirk Slama,*
  Enterprise SOA: Service-Oriented Architecture Best Practices,
  September 2006, Prentice Hall
- *Gero Mühl, Ludger Fiege, Peter R. Pietzuch,*
  Distributed Event-Based Systems,
  Juli 2006, Springer Verlag
- *Shrideep Pallickara, Geoffrey Fox*,
  NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids
  June 2003, ACM/IFIP/USENIX International Middleware Conference Middleware

Backup

Automatisierte Messdaten Erfassung



- Phase 1 (*experiment design*)
  - XML-like configuration file generation
  - Define experiment parameters (duration, …)
- Phase 2 (*measurement*)
  - Automated measurement execution and logging (set of shell and perl scripts)
  - Validation of collected data (conform to constraints?)
- Phase 3 (*evaluation*)
  - Consistent measurement data storage
  - Analysis and visualisation with MatLab

Verschiedene JMS Server - Übersicht

FioranoMQ	Commercial	Measured	Modelled
SunMQ	Commercial	Measured	Modelled
WebsphereMQ	Commercial	Measured	Modelled
Bea WebLogic	Commercial	Measured	Modelled
ActiveMQ	Open Source	Measured	Modelled
JBoss	Open Source	Installed	-
OpenJMS	Open Source	Installed	-

- Aktueller Stand der Untersuchungen verschiedener JMS Server Produkte

Server Cluster

- Verschiedene Topologien für Cluster
  - Hub-and-Spoke
  - Bus
  - Mesh
  - P2P (NaradaBrokering)

Keine dieser Topologien skaliert bezüglich des Nachrichtendurchsatzes