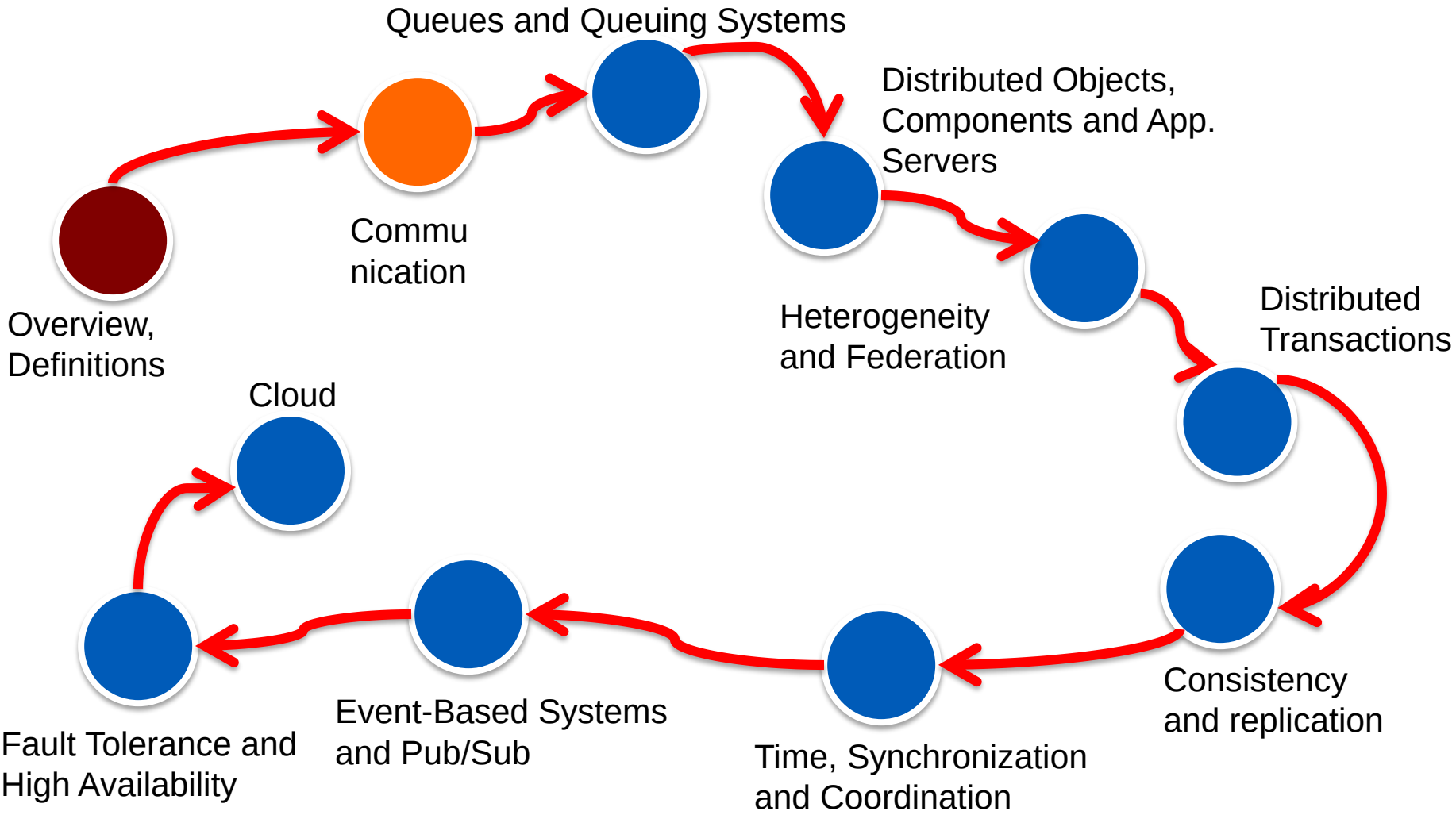
Middleware:
2. Communication

A. Buchmann
Wintersemester 2013/2014

TECHNISCHE UNIVERSITÄT DARMSTADT

DVS

Topics


Queues and Queuing Systems
Distributed Objects, Components and App. Servers
Commu nication
Overview, Definitions
Heterogeneity and Federation
Distributed Transactions
Cloud
Consistency and replication
Fault Tolerance and High Availability
Event-Based Systems and Pub/Sub
Time, Synchronization and Coordination

Topics

TECHNISCHE UNIVERSITÄT DARMSTADT

- Networking in brief
- Interaction modes

- Client-Server Communication

- Group Communication

- Message + Message Semantics
- Message-oriented communication

- Naming and Directory Services

Reading for THIS Lecture

TECHNISCHE UNIVERSITÄT DARMSTADT

- The slides for the lecture are based on material from:
  - George Coulouris, Jean Dollimore, and Tim Kindberg. 2005. **Distributed Systems: Concepts and Design.** Addison-Wesley Longman.
    - Chapter 3 (overview), Chapter 4
  - Andrew S. Tanenbaum and Maarten Van Steen. 2001. **Distributed Systems: Principles and Paradigms.** Prentice Hall.
    - Chapter 2
  - Christof Leng, Wesley Terpstra, Max Lehn, Robert Rehner, Alejandro Buchmann

    All-Weather Transport Essentials, IEEE Internet Computing, Nov/Dec 2012

Networking Basics in Brief

TECHNISCHE UNIVERSITÄT DARMSTADT

DVS

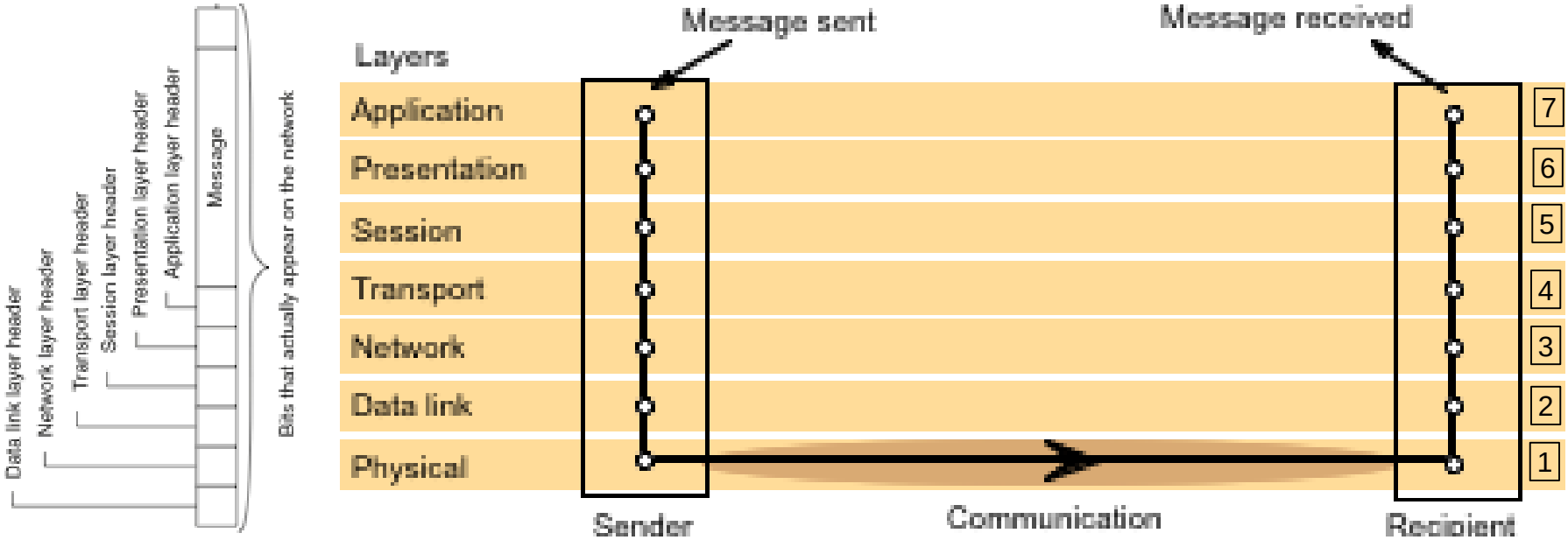
Remember from other courses

TECHNISCHE UNIVERSITÄT DARMSTADT

- Due to absence of shared global state (shared memory)
  - Processes communicate → Message exchange
- Packet switching
  - Packets for different destination share communication link
- Two basic transmission modes:
  - Packet Transmission
    - App. → Messages → Packets → Queue (buffer) → Asynchronous transmission
  - Data Streaming
    - Streaming Audio | Video
    - Guaranteed rates: Bandwidth, bounded latencies, link availability
    - Establish channel – predefined route
    - Use Buffering – mask delays

Protocols

- Protocol – a set of rules and formats used for communication
  - Specification of sequence of messages
  - Specification of message (data) format and semantics
  - (optional) Specification of failure handling and compensation
- Networking protocols have layered structure (protocol stack)



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design

Important Layers

TECHNISCHE UNIVERSITÄT DARMSTADT

- Physical layer
  - specification and implementation of bits, and their transmission
- Data link layer
  - transmission of a series of bits into a frame, allows error and flow control

- Network layer
  - **routing** of packets
  - example: connectionless IP (packets routed independently, different routes)

- **Distributed systems start from network layer upwards**

- Transport Layer
  - provides the actual communication facilities for most distributed systems
    - **reliable transport connections**, packet sequencing and assembly→end-end comm.
  - Standard Internet protocols
    - TCP: connection-oriented, reliable, stream-oriented communication
    - UDP: unreliable (best-effort) datagram communication

OSI protocol summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- For distributed systems: the lowest level → network layer

Layer	*Description*	*Examples*
Application	Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service.	HTTP, FTP , SMTP, CORBA IIOP
Presentation	Protocols at this level transmit data in a network representation that is independent of the representations used in individual computers, which may differ. Encryption is also performed in this layer, if required.	Secure Sockets (SSL),CORBA Data Rep.
Session	At this level reliability and adaptation are performed, such as detection of failures and automatic recovery.	
Transport	This is the lowest level at which messages (rather than packets) are handled. Messages are addressed to communication ports attached to processes, Protocols in this layer may be connection-oriented or connectionless.	TCP, UDP
Network	Transfers data packets between computers in a specific network. In a WAN or an internetwork this involves the generation of a route passing through routers. In a single LAN no routing is required.	IP, ATM virtual circuits
Data link	Responsible for transmission of packets between nodes that are directly connected by a physical link. In a WAN transmission is between pairs of routers or between routers and hosts. In a LAN it is between any pair of hosts.	Ethernet MAC, ATM cell transfer, PPP
Physical	The circuits and hardware that drive the network. It transmits sequences of binary data by analogue signalling, using amplitude or frequency modulation of electrical signals (on cable circuits), light signals (on fibre optic circuits) or other electromagnetic signals (on radio and microwave circuits).	Ethernet base- band signalling, ISDN

Middleware protocols

TECHNISCHE UNIVERSITÄT DARMSTADT

- Middleware is conceptually on lower portion of application layer
  - for performance reasons it may go down to transport or network layer
    - Example: reliable multicast

- Middleware provides common services and protocols to different applications:

  - High level communication protocols →
    - → Distributed commit protocols → atomicity
    - → Distributed locking protocols → concurrency, shared resources

  - Protocols for middleware services → Authentication, Replication protocols

Basic Network Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

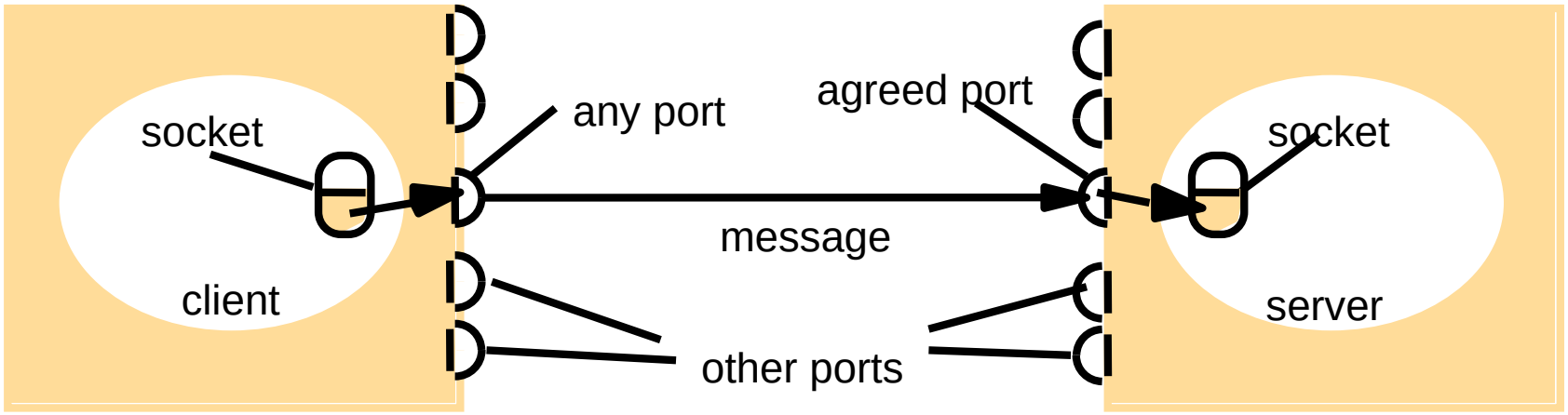
Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Sockets and Ports

TECHNISCHE UNIVERSITÄT DARMSTADT

- IPC, TCP and UDP use Sockets as an abstraction → comm. Endpoint
  - Receive Msg: Associate socket to a local port and an Address
  - Send Msg: send to an address and port
  - Possible ports: 64K, not shared between processes
- Blocking: non-blocking Send(msg), blocking Receive(Msg)
- Failure model: Omission failures (dropped msg), Ordering (out of order)



DVS

UDP and TCP

TECHNISCHE UNIVERSITÄT DARMSTADT

- UDP:
  - simple message passing facility
  - no acknowledgements or retries
  - omission failures
  - no built-in performance penalties

- TCP:
  - guarantees message delivery (ack scheme)
  - flow control (block sender until receiver consumed sufficient data)
  - Message identifiers allow the receiver to eliminate duplicates and reorder messages that arrived out of order
  - expense of additional messages, higher latency and storage costs

UDP Datagrams

TECHNISCHE UNIVERSITÄT DARMSTADT

- Message size:
  - the receiving process needs to specify an array of bytes of particular size in which to receive the message
- Blocking
  - non-blocking sends and blocking receives
  - messages are discarded at the destination if no process already has a socket bound to the destination port
- Failure model
  - omission failures
  - ordering
  - using checksum
- Use of UDP
  - used if occasional omission failures are acceptable
  - attractive because does not suffer from overhead associated with guaranteed message delivery

Example: UDP Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

- UDP Client:
  - DatagramPacket request = new DatagramPacket(m, args[0].length(), aHost, serverPort);
  - aSocket = new DatagramSocket(); aSocket .send(request);
  - …
  - byte[] buffer = new byte[1000];
  - DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
  - aSocket.receive(reply);

- UDP Server:
  - aSocket = new DatagramSocket(6789);
  - byte[] buffer = new byte[1000];
    - while(true){
      - DatagramPacket request = new DatagramPacket(buffer, buffer.length);
      - aSocket.receive(request);
      - DatagramPacket reply = new DatagramPacket(request.getData(),
      - request.getLength(), request.getAddress(), request.getPort());
      - aSocket.send(reply);
    - }

TCP Streams

TECHNISCHE UNIVERSITÄT DARMSTADT

- Message sizes: application-specific
- Lost messages: uses an acknowledgement scheme
- Flow control: attempts to match the speeds of the processes
- Message destinations
  - a pair of processes establish a connection before they can communicate over a stream
- Use of sockets
  - the pair of sockets in a client and a server are connected by a pair of streams, one in each direction
- Failure model
  - use checksums to detect and discard corrupt packets
  - use timeouts and re-transmissions to deal with lost packets

Classification: Modes of Interaction and their Implementation

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Modes of Interaction: General Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

Initiator of Interaction	Knowledge about Counterpart: Full	Knowledge about Counterpart: No
Consumer	**Request/Reply** *Implementations*: *RPC, ROI/RMI, HTTP Comm.*	**Anonymous Request/Reply** *Implementations*: *Load Balancing,* *Mediated web services* *File sharing*
Producer	**Point-to-Point** **(peer-to-peer) Message based** *Implementations*: *Unicast, Group Communication, eMail-List*	**Event-based Dissemination** *Implementations*: *Publish/Subscribe Notification*

Modes of Interaction

TECHNISCHE UNIVERSITÄT DARMSTADT

- 2 dimensional problem space: classification of interaction types
  - Separate concept from implementation (e.g. Request/Reply vs. RPC)

- **Knowledge about Counterpart**
  - Full – the process knows identity, address, protocol and message format of counterpart
  - No – concrete counterpart is unknown (assumption: counterpart exists)
    - Counterpart Discovered – middleware mediation
    - Message delivered to counterpart according to some criteria (load, capabilities, …)

- **Initiator of Interaction**
  - Who initiates the communication?
  - Roles: Consumer, Producer

Request/Reply: RPC, ROI/RMI, …

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

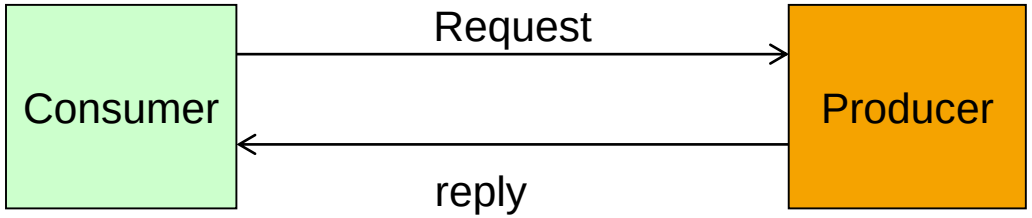
DVS

Request/Reply

Initiator	K. Counterpart: Full	K. Counterpart: No
C		
P		

TECHNISCHE UNIVERSITÄT DARMSTADT

- Direct and synchronous communications
  - Enforces tightly coupling of comm. parties
  - Impairs scalability → blocking (client waits for response)
  - Need for asynchronous and decoupled operations
  - **Representatives**: IPC, RPC, ROI/RMI, HTTP Request/Response



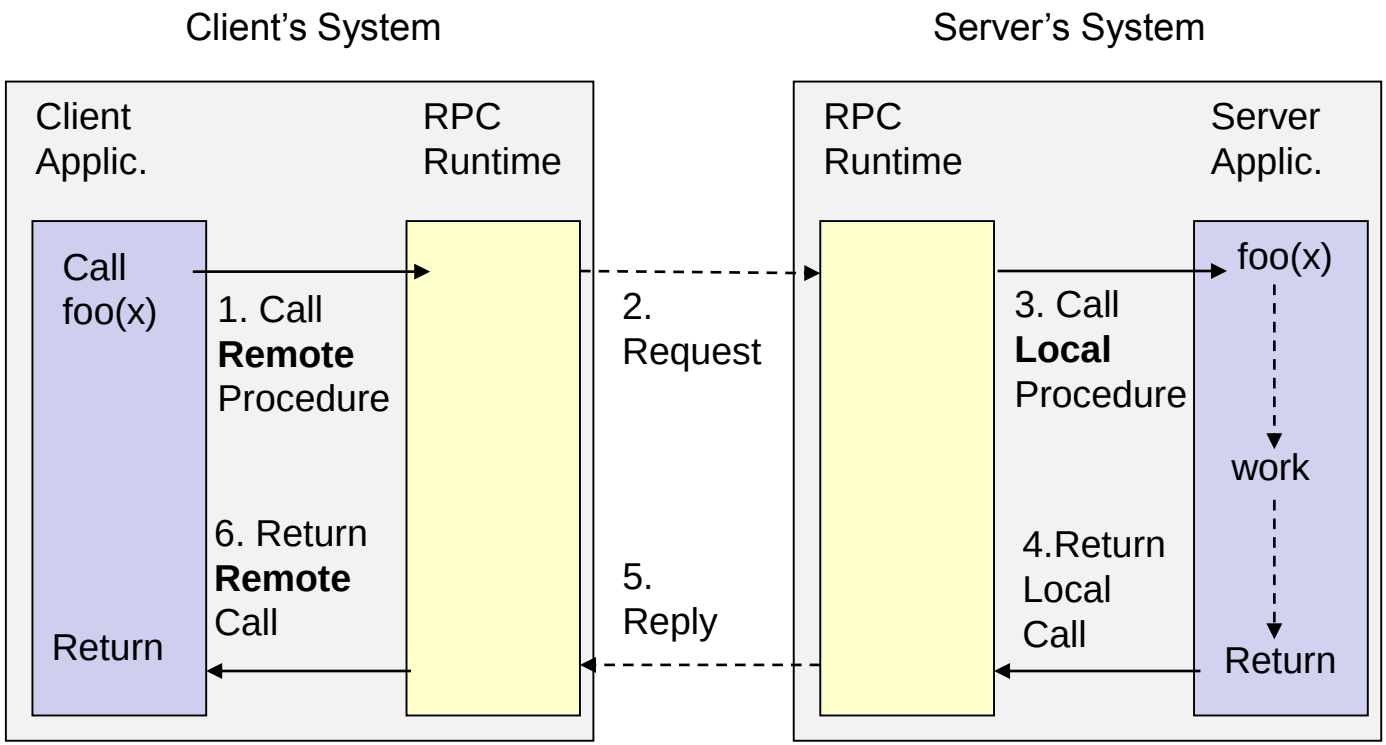
Known Server:
Address, Protocol,
Request type

- Polling: clients “pull” data from remote data sources
  - Trade off: polling cycle vs. accuracy of changing data/situation
    - Short polling interval → waste resource
    - Long polling interval → decrease freshness, risk missing situation

DVS

Basic RPC Model

- Simple: Remote procedure calls mimic behavior of local procedure calls
  - Many alternatives: DCE RPC, Asynchronous RPC, Transactional RPC, …



Basic RPC Model

TECHNISCHE UNIVERSITÄT DARMSTADT

- Simple: Remote procedure calls mimic behavior of local procedure calls
  - Transparency:
    - Remote calls treated as local ones
    - Communication details hidden → procedure call abstraction
      - Hide distribution
    - Programming Language transparency
    - Hardware transparency
  - Well-designed procedures operate in isolation
- Safe: every call receives exactly one return
  - return from called procedure or
  - exception from exception handler
- No sequencing of messages required
- Synchronous
  - calling process stops, waits for procedure to execute and control to be returned

Transparency: Client and Server Stubs

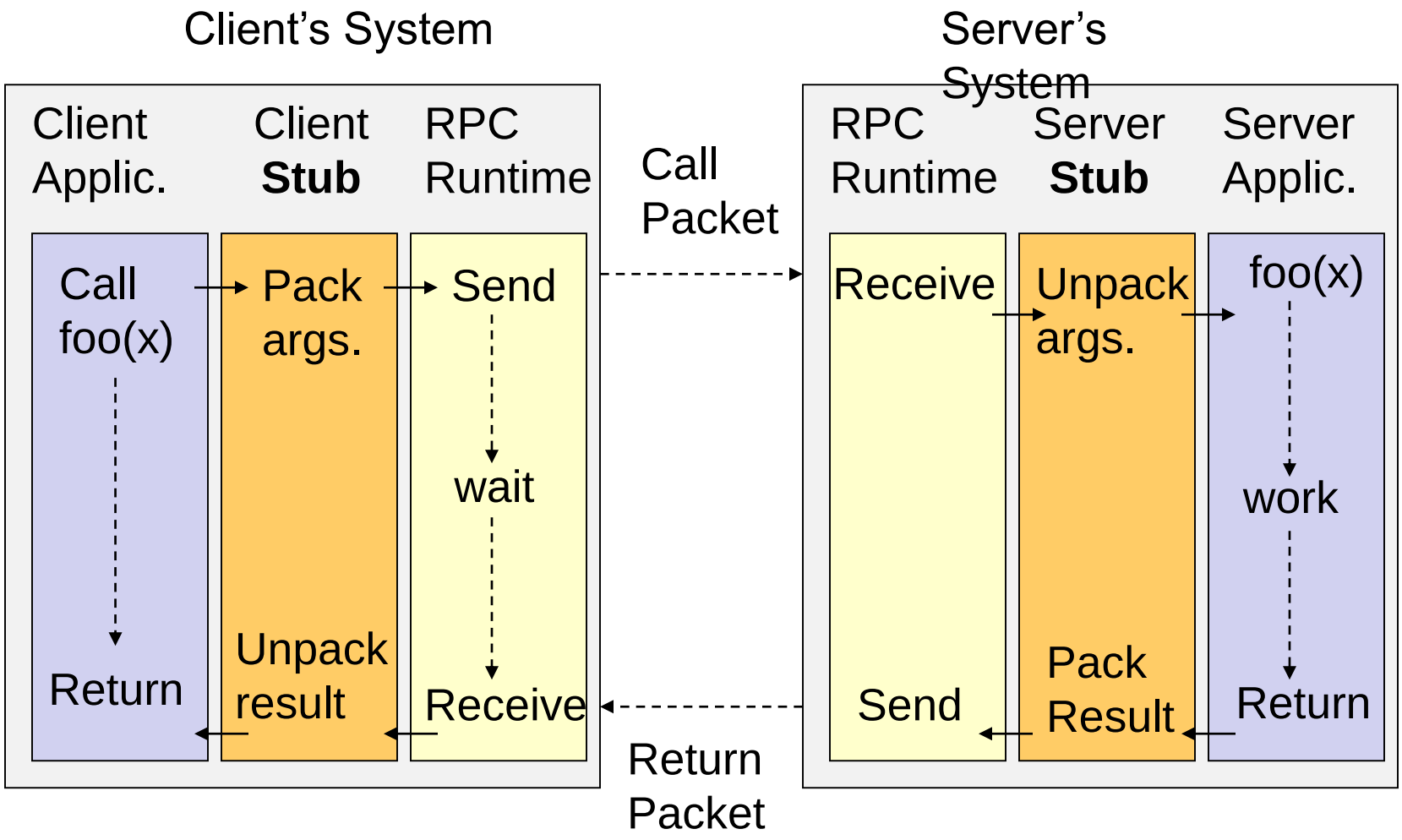
TECHNISCHE UNIVERSITÄT DARMSTADT

- Transparency mostly realized through:
  - Stubs
  - Interface definition

- Stubs – local representatives of the counterpart, generated from Interface Definition
  - Implement parameter passing. Perform Un/Blocking. Handle Failures
- Client Stub → Pack call parameters in Message, Send to server.
- Server Stub → Listen for requests. Unpack and perform local calls.
  - Pack reply, send back reply

- Different approaches - different terminology
  - Stub → Stub, Stub → Skeleton, Proxy → Stub
  - (RPC, RMI, CORBA, .NET, JEE)

RPC Operation



Parameter Passing

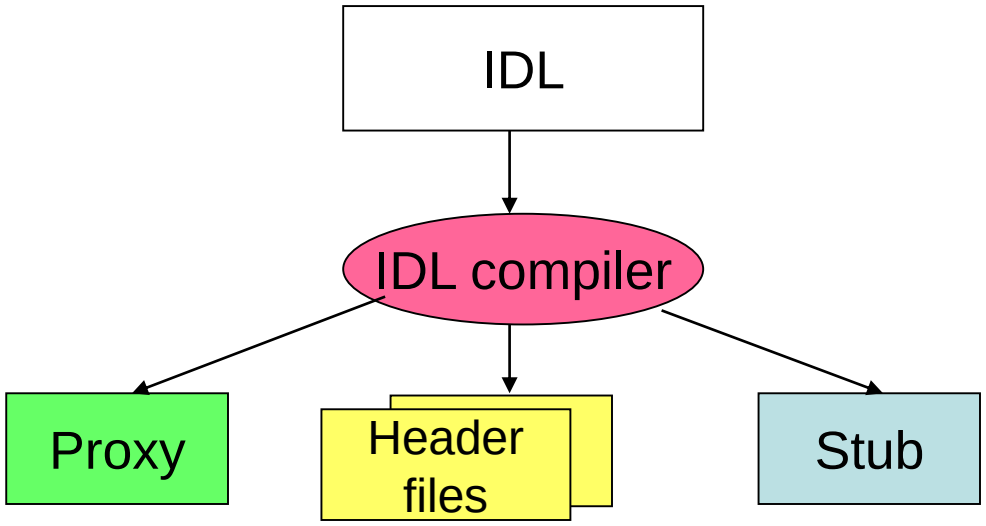
TECHNISCHE UNIVERSITÄT DARMSTADT

- Packing/unpacking parameters in a message–Marshalling/Unmarshalling
  - Heterogeneity →Uniform representation:
    - Hardware, Programming Languages, Encodings, Network Protocols
    - Strategies: standard format (CDR), translate to server format, receiver-makes-it-right
- Value Parameters
  - Hardware – Big Endian (e.g. Intel), Little Endian (SPARC, PPC)
    - Big Endian = 0xAABBCCDD | Little Endian = 0xDDCCBBAA
  - Programming Languages
  - Character Encodings
- Reference Parameters and complex data structures
  - Forbid ‘pass-by-reference’
  - Flatten structures → copy and flatten (Java object serialization)
  - Use lengths and separator characters

Interface Specification and Stub Generation

TECHNISCHE UNIVERSITÄT DARMSTADT

- IDL compiler produces
  - header files to be included in calling program
  - proxy procedures that are linked with caller
  - stub procedures that are linked with the server

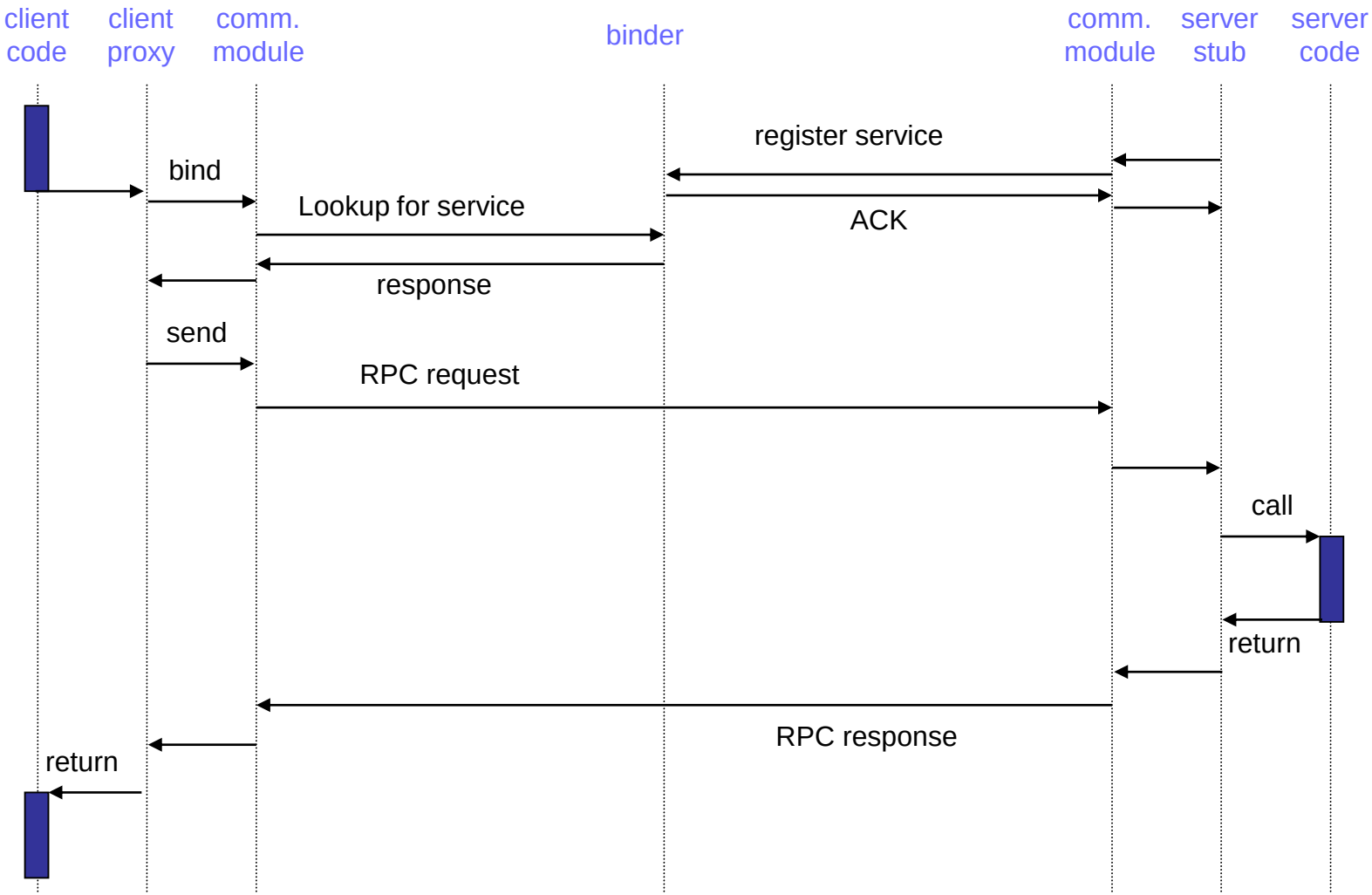


DCE RPC: Binding Clients and Servers
(DCE = Distributed Computing Environment)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In Distributed environments:
  - Client must (1) locate server machine, and (2) locate the server.
- Static Binding:
  - Remote procedure must be referenced at compile time
    - Write interface definition for called program
    - Interface definition specifies name, type and arguments of procedure
    - Interface definition processed by interface definition language (IDL) compiler (stub compiler)
- Communication Binding
  - Communication binding between client and server must be established
  - Server must export/register its interface
    - what interface is supported
    - where it can be located
  - Client creates communication based on exported interface
  - Problem: how do clients know where the service in question is located?
- → Naming and Directory Services

DCE RPC: Sequence Diagram



Fault tolerance in RPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- RPC is an end-to-end protocol that supports the interaction between two entities
- Termination is always at initiator, no parallel calls
- When there are more entities interacting with each other, RPC treats the calls as independent of each other (however, these calls are not independent)
- Recovering from partial system failures is very complex
- Avoiding these problems using plain RPC is very cumbersome

- Idempotent vs. non-idempotent servers
- If client doesn't receive answer, was the procedure call or return message lost?
  - Resend to idempotent server
  - Inquire from non-idempotent server
- Semantics
  - Maybe or best effort (no guarantee)
  - at least once (idempotent operation)
  - at most once (non-idempotent operation)
  - exactly once (committing transaction)

Performance of RPC

TECHNISCHE UNIVERSITÄT DARMSTADT

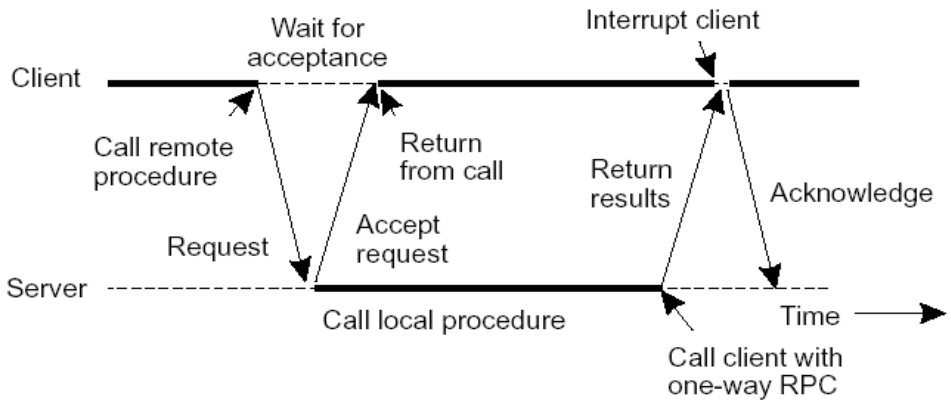
- 3 main components
  - marshaling/unmarshaling of parameters
  - RPC runtime and communication software
  - physical network transfer
- Total cost of one RPC ~ 10 000 -15 000 machine instructions
- Local PC approx. 100 times faster!

Overview: Asynchronous RPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Traditional RPC is synchronous, i.e. strict Request/Reply behavior
  - Client blocked for the duration of the call → not scalable

- Asynchronous RPC behavior
  - Client performs a Call and continues
  - Upon Return the Client is notified



DVS

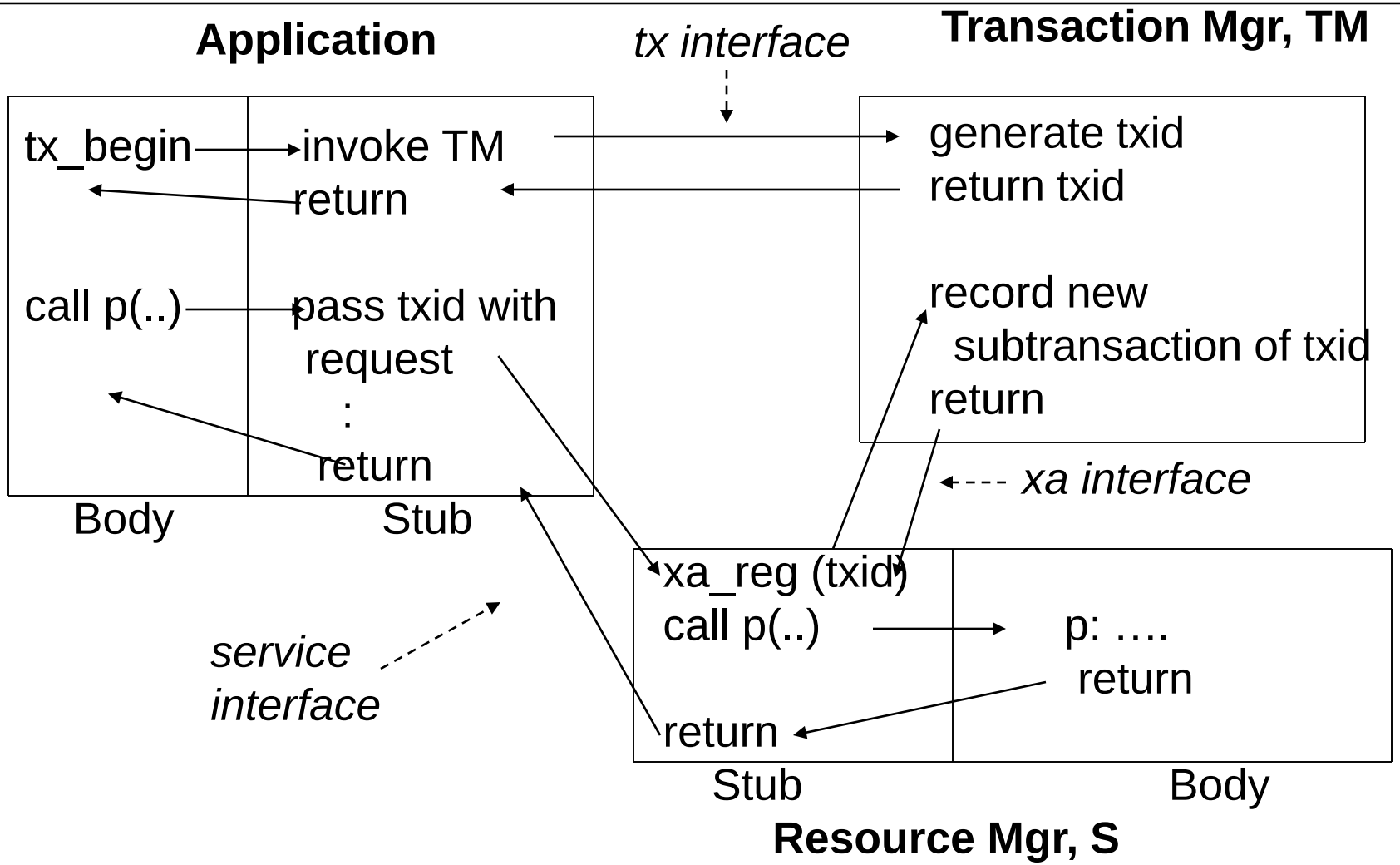
Overview: Transactional RPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Very Very Brief Overview (transactions later):

- Coupling of RPC with transactions
  - transactional messages between C & S
  - atomic procedures (all or nothing)
  - results are made visible through commit
- Additional data must be passed
  - transaction ID
  - coordination information (2 phase commit)
  - name and location of calling TP system
  - exception codes

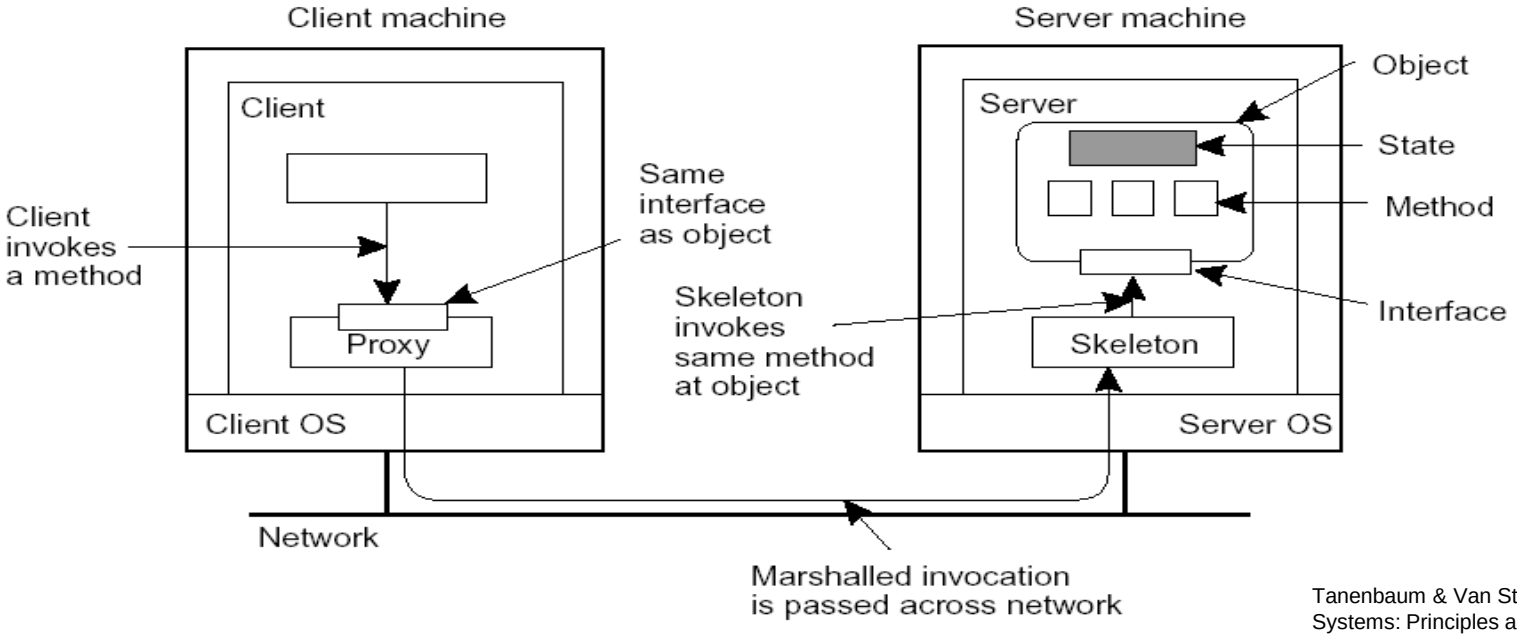
Overview: Transactional RPC (more later)



Overview: Remote Object/Method Invocation (details later)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Data and operations encapsulated in an object
- Operations = methods, and are accessible through interfaces
- Object offers only its interface to clients
- Object server is responsible for a collection of objects
- Client stub (proxy) implements interface
- Server skeleton handles (un)marshaling and object invocation



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms

DVS

Point-to-Point Messages (also known as Peer to Peer, P2P), Group Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

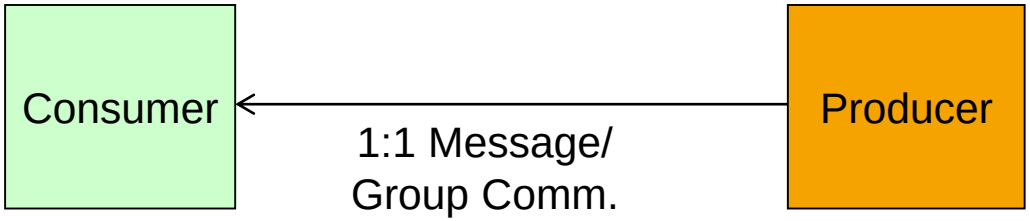
DVS

Point-to-Point Communication (message passing)

Initiator	K. Counterpart: Full	K. Counterpart: No
C		
P		

TECHNISCHE UNIVERSITÄT DARMSTADT

- Consumer of message is known to producer
- Producer sends message directly to consumers
  - manage list of interests and registered consumers



Known Client(s):
Address, Protocol, Request type
Server Initiates:
communication

DVS

Comparison RPC vs. Point-to-Point

TECHNISCHE UNIVERSITÄT DARMSTADT

- Four areas of comparison:
  - Flexibility of allowed message sequences
  - Termination model for transactions
  - Connection model, i.e. how the state of transactions spanning multiple programs is handled
  - Effect of communication mechanism on application design

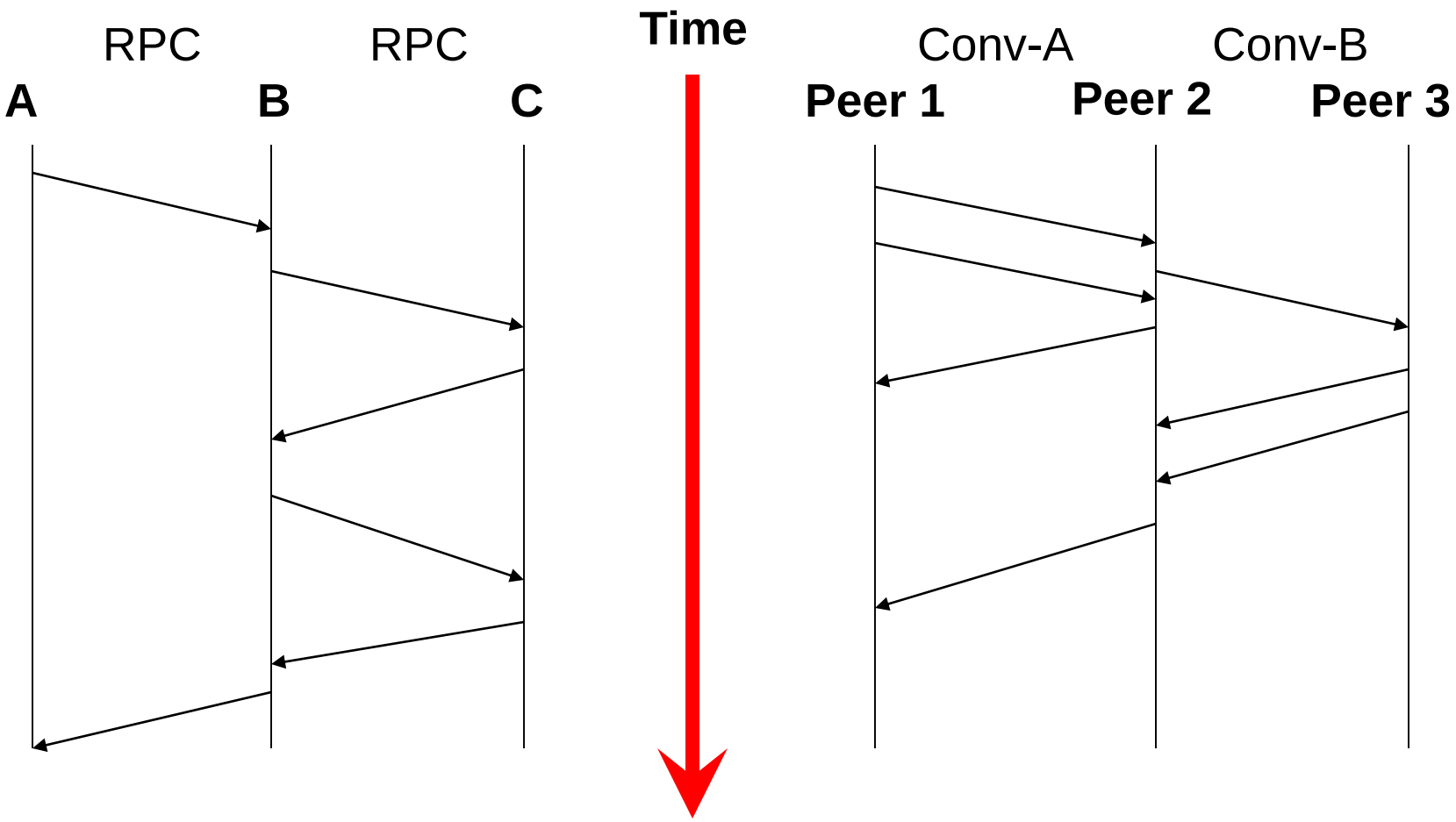
- Programming
  - RPC
    - much simpler
    - same call-return semantics in local and remote procedure calls
    - hard-wired termination, simple semantics
  - Point-to-Point
    - communication mechanism reflected in application logic
    - timing information (waiting for messages) part of program
    - More flexible, one way messages (termination at server possible), parallelism

RPC vs. Point-to-Point: Messages

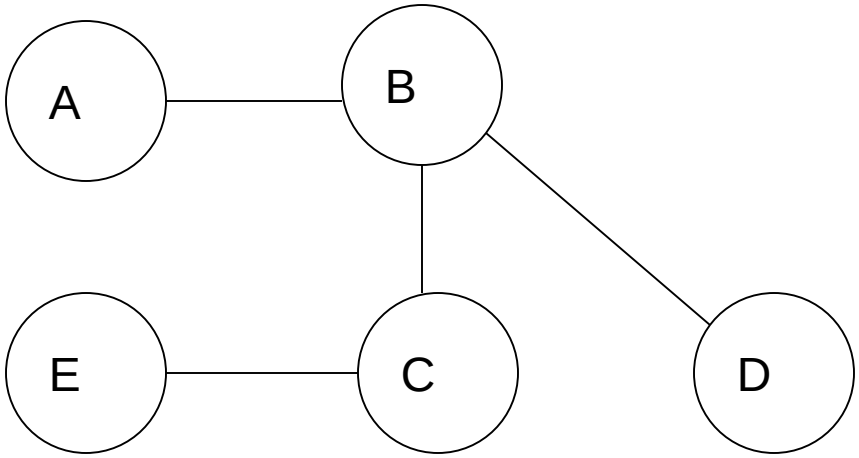
TECHNISCHE UNIVERSITÄT DARMSTADT

- RPC only allows call-return style with matched-up messages
- Point-to-Point more flexible
  - special case can emulate call-return
  - multiple messages possible before receiving return
    - higher parallelism (messages to multiple programs)
    - no blocking while waiting for return
    - message batching
- Point-to-Point more complex and error-prone but often used at low system levels for performance reasons

RPC vs. Point-to-Point: Msg.



Peer-To-Peer Communication



- A module can have multiple connections concurrently
  - Each connection associated with transaction that created it
  - A new instance of called program is created when a connection is established
  - The connections associated with a particular transaction form an acyclic graph
  - All nodes of graph coequal (no root as in RPC)

Peer-To-Peer and Commit

TECHNISCHE UNIVERSITÄT DARMSTADT

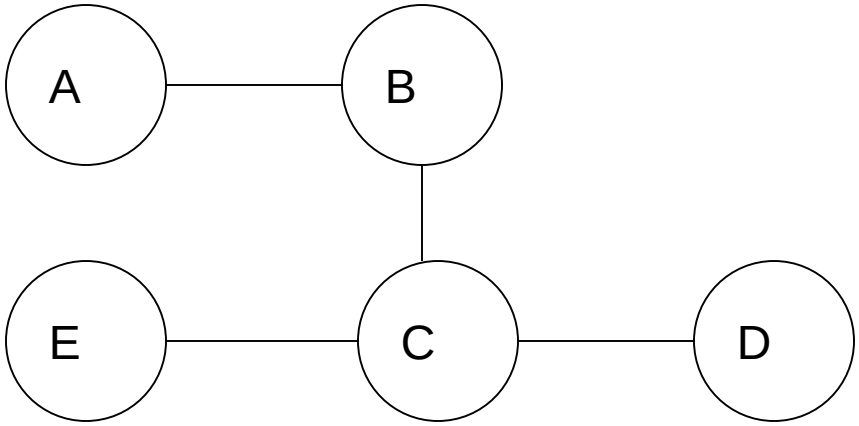
- Problems:
  - Coequal status: Since there is no root (as in RPC) who initiates the commit?
  - Asynchronous: How can initiator of commit be sure that all nodes have completed their computations?
    - This is not a problem with RPC.

Solution: Syncpoints in Half Duplex Systems

TECHNISCHE UNIVERSITÄT DARMSTADT

- One node, A, initiates commit. It does this when
  - it completes its computation and
  - all its connections are in send mode.
- A declares a syncpoint and waits for protocol to complete.
- Transaction manager sends syncpoint message to all neighbors, B.
- When B completes its computation and all its connections (other than connection to A) are in send mode, B declares syncpoint and waits.
- Transaction manager sends syncpoint message to all B's neighbors (other than A).
- When syncpoint message reaches all nodes, all computation is complete and all have agreed to commit.

Syncpoint Error



- Problem: Two nodes, all of whose connections are in send mode, initiate commit concurrently
- Result: Protocol does not terminate. First node to receive two syncpoint messages cannot declare a syncpoint

RPC vs. Point-to-Point: Termination

TECHNISCHE UNIVERSITÄT DARMSTADT

- RPC:
  - called program terminates by returning to caller
  - transaction termination always initiated by caller

- Point-to-Point:
  - called programs may initiate commit
  - commit may be initiated by one program while others have not finished
  - SYNCPOINT rules delay commit until all finished
  - no active programs are committed but deadlock possible

RPC vs. Point-to-Point: Connection

TECHNISCHE UNIVERSITÄT DARMSTADT

- Point-to-Point connection oriented:
  - direction of communication
  - direction of the link (i.e. send or receive mode)
  - ID and state of the transaction (active, committed, aborted) and of individual programs (e.g. cursor)
  - connection state not recoverable
- RPC connectionless communication model:
  - client and server don’t share state
  - explicit passing of state back and forth
  - context handle provided by some implementations

RPC vs. Point-to-Point: Trends

TECHNISCHE UNIVERSITÄT DARMSTADT

- General application programming: strong trend towards RPC due to simplicity

- System software (DBMS, TPM, etc.) Point-to-Point messaging due to flexibility and higher efficiency (parallelism and suppression of unnecessary return messages)
  - exception: Oracle SQL*Net uses RPC

Unicast vs. Broadcast vs. Multicast

TECHNISCHE UNIVERSITÄT DARMSTADT

- Unicast is unidirectional 1:1 messaging
- Broadcast sends message to all participants
  - e.g. Ethernet
- Multicast sends same message to a group of participants
  - identification of group membership based on
    - physical addresses
    - content of message (subject-based addressing)
- ANYCAST – does a recipient exist

Unicast

TECHNISCHE UNIVERSITÄT DARMSTADT

- Unidirectional point to point data transfer
- Control information may flow in both directions (acks, nacks)
  - letter is data transfer
    - if sent with return receipt, receipt is just control info but not a new data transfer
  - answer letter is new data transfer
- Unicast doesn’t scale for large groups

Group Communication: Multicast

TECHNISCHE UNIVERSITÄT DARMSTADT

- Same message sent by one process to a group of processes
- Useful infrastructure for fault tolerance in distributed applications
- Different options (quality of service):
  - unreliable multicast – same failure model as UDP datagrams
  - reliable multicast
  - atomic multicast

Reliability in Multicast

TECHNISCHE UNIVERSITÄT DARMSTADT

- Unreliable Multicast
  - Best effort message delivery
  - No guarantees are given
  - Acceptable mostly for non-critical messages → load info
  - Used by some data dissemination applications
- (k)-Reliable Multicast
  - Reliable multicast provides guaranteed delivery to (at least k) receivers
  - Reliability depends on failure model
  - As k grows, cost increases
  - Different qualities of service possible
- Atomic Multicast
  - Message is received by all members of group or by none
  - Failed processes are excluded from a group

Ordering in Multicast

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ordering specifies the sequence in which messages are sent and delivered/received
  - FIFO - based on single sender
  - total ordering - requires centralized control
  - causal ordering - establishes partial orders over mutually relevant messages

Anonymous (mediated) Request/Reply

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

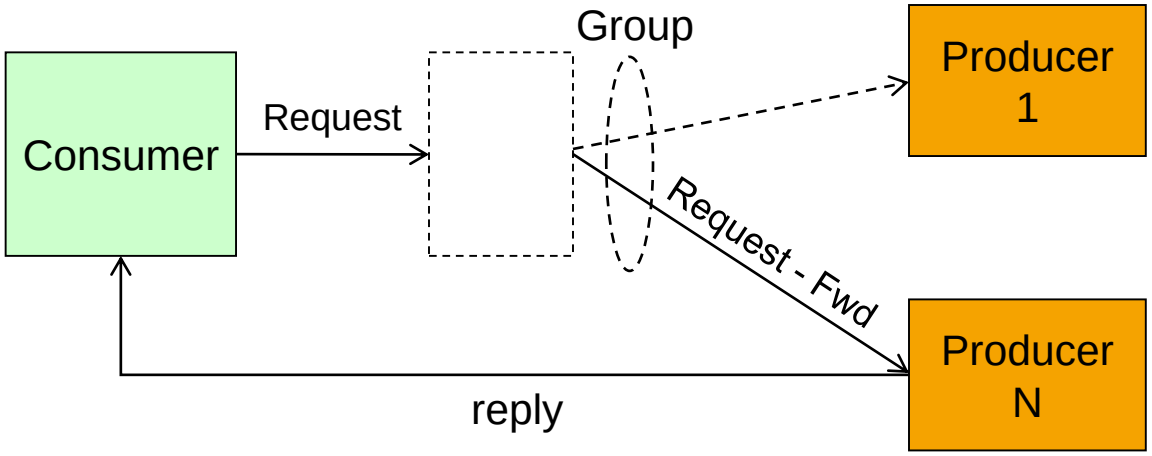
DVS

Anonymous Request/Reply

Initiator	K. Counterpart: Full	K. Counterpart: No
C		
P		

TECHNISCHE UNIVERSITÄT DARMSTADT

- Consumer does not specify the provider
- Request is delivered to a selected provider out of s set of providers
  - a single provider out of a set of possible, equal providers
  - the selected provider processes the request and delivers the reply to the client
- Identity of provider is a priori unknown to requestor
- Examples: Load balancing, Method Invocation on an abstract interface, Facades, Web Services with UDDI mediation



Unknown:
Provider address
Known
Protocol, Request type, facade

DVS

Event-based Interaction Publish/Subscribe

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

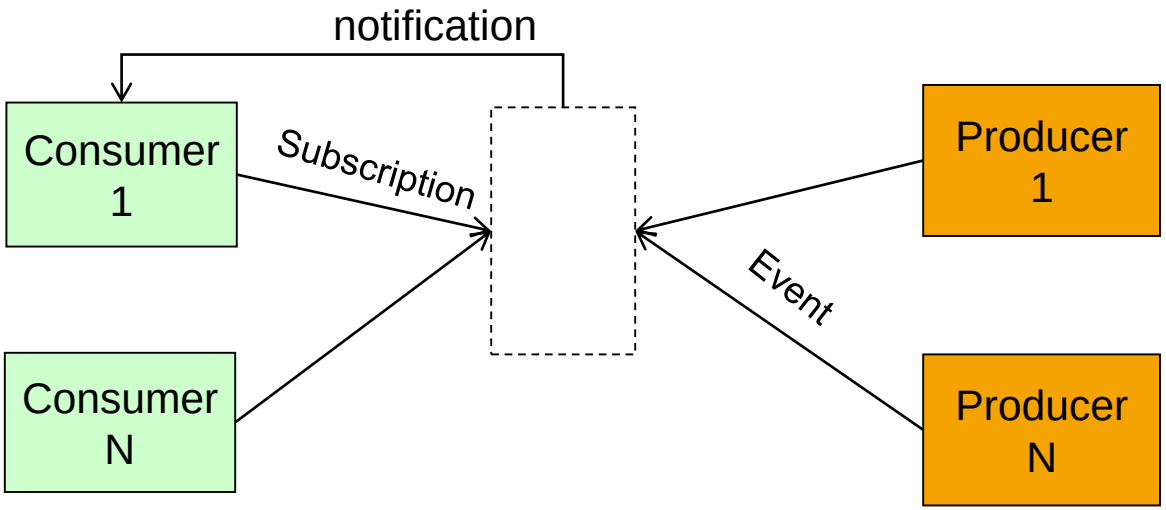
DVS

Event-Based Dissemination

Initiator	K. Counterpart: Full	K. Counterpart: No
C		
P		

TECHNISCHE UNIVERSITÄT DARMSTADT

- The producer is the initiator of a concrete communication
- Consumers issue subscriptions (interests)
  - Consumers are not aware of producers
- Notifications are addressed to a consumer based on the subscriptions managed by the (publish/subscribe) middleware
  - Producers are not aware of specific consumers
- Notifications are delivered to consumers if they match with subscriptions
- → Topic 7



DVS

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Networking Basics – brief summary
  - Networking layers, Basic mechanisms
- Modes of Interaction
- Request Reply
  - RPC as most common example
- Point-to-Point Messaging
  - Message passing, unicast, multicast
  - Group Communication
- Anonymous Request Reply
- Event-Based Dissemination

Thank You!

TECHNISCHE UNIVERSITÄT DARMSTADT

Questions?

DVS