IF3230 – Sistem Paralel dan Terdistribusi
Communication

Achmad Imam Kistijantoro (imam@informatika.org)
Judhi Santoso (judhi@informatika.org)
Anggrahita Bayu Sasmita (bayu.anggrahita@informatika.org)

Communication

- Interprocess communication → the heart of all distributed system

- Communication system in distributed system is always based on **low-level message passing** as offered by the underlying network

Low level layer

- Untuk sebagian besar sistem terdistribusi, antarmuka terbawah yang digunakan adalah layer network
- Transport layer: transport layer yang sesungguhnya menyediakan fasilitas komunikasi untuk sebagian besar sistem terdistribusi.
- Standard protocol: TCP, UDP

Middleware layer

- Middleware adalah layer yang dibuat untuk menyediakan layanan dan protokol umum yang dapat digunakan berbagai aplikasi.
  - Protokol komunikasi yang beragam
  - (un)marshaling data
  - Naming protocols => memudahkan sharing resources
  - Security protocols => untuk secure communication
  - Scaling mechanisms => seperti replikasi dan caching

  - Sehingga yang tersisa untuk dikembangkan adalah protokol spesifik aplikasi

Type of Communication

- Transient vs Persistent communication

- Transient
  - Message tidak disimpan (discard jika tidak berhasil dikirim)
  - Aplikasi sender & receiver harus aktif saat proses pengiriman message

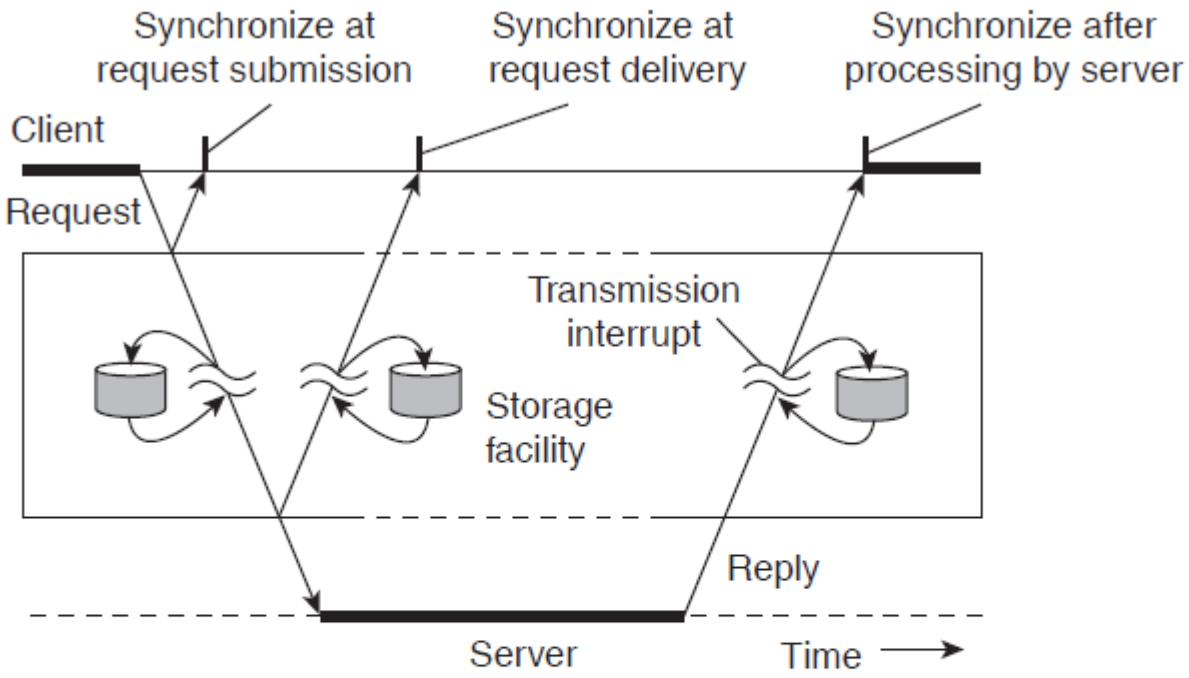
- Persistent
  - Message disimpan oleh communication middleware sampai message dapat dikirim ke receiver
  - Apllikasi sender maupun receiver tidak harus aktif pada saat bersamaan

Type of Communication

- Asynchronous vs synchronous communication

- Asynchronous
  - Sender melanjutkan proses tanpa menunggu ACK (non-blocking)
- Synchronous
  - Sender blocking hingga ada informasi ketersampaian message

Synchronous Communication



Places for synchronization

- At request submission
- At request delivery
- After request processing

Client/Server

Some observations

Client/Server computing is generally based on a model of transient synchronous communication:

- Client and server have to be active at time of commun.
- Client issues request and blocks until it receives reply
- Server essentially waits only for incoming requests, and subsequently processes them

Drawbacks synchronous communication

- Client cannot do any other work while waiting for reply
- Failures have to be handled immediately: the client is waiting
- The model may simply not be appropriate (mail, news)

Models for Communication

- 3 widely-used models for communication
  - Message-Oriented Communication
  - Remote Procedure Call (RPC)
  - Stream-Oriented Communication

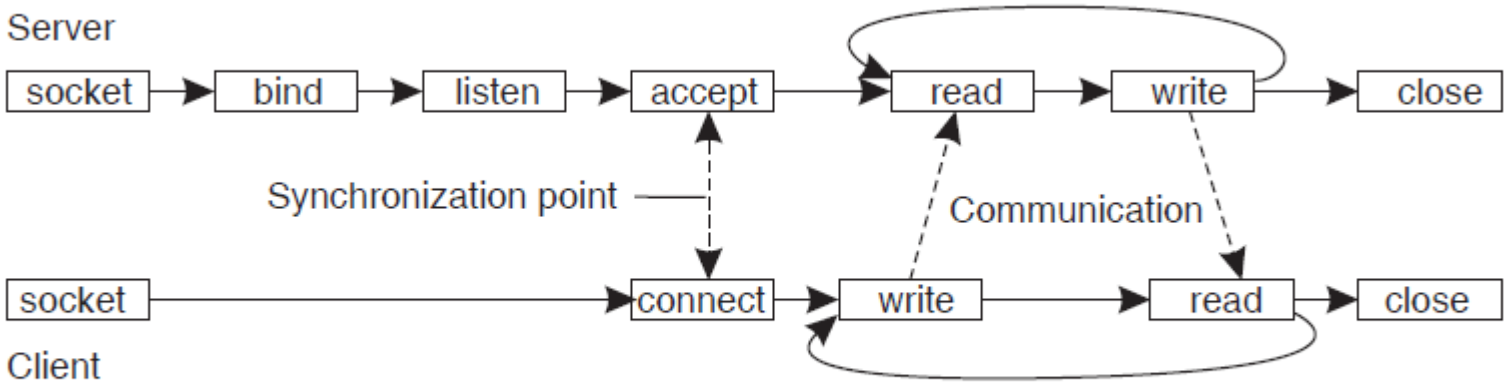
Message Oriented Communication

Message Oriented Communication

- Message-oriented transient communication
- Message-oriented persistent communication (Message Queueing System / Message Oriented Middleware)

Message-oriented Transient Communication

- Simple message-oriented model offered by transport layer

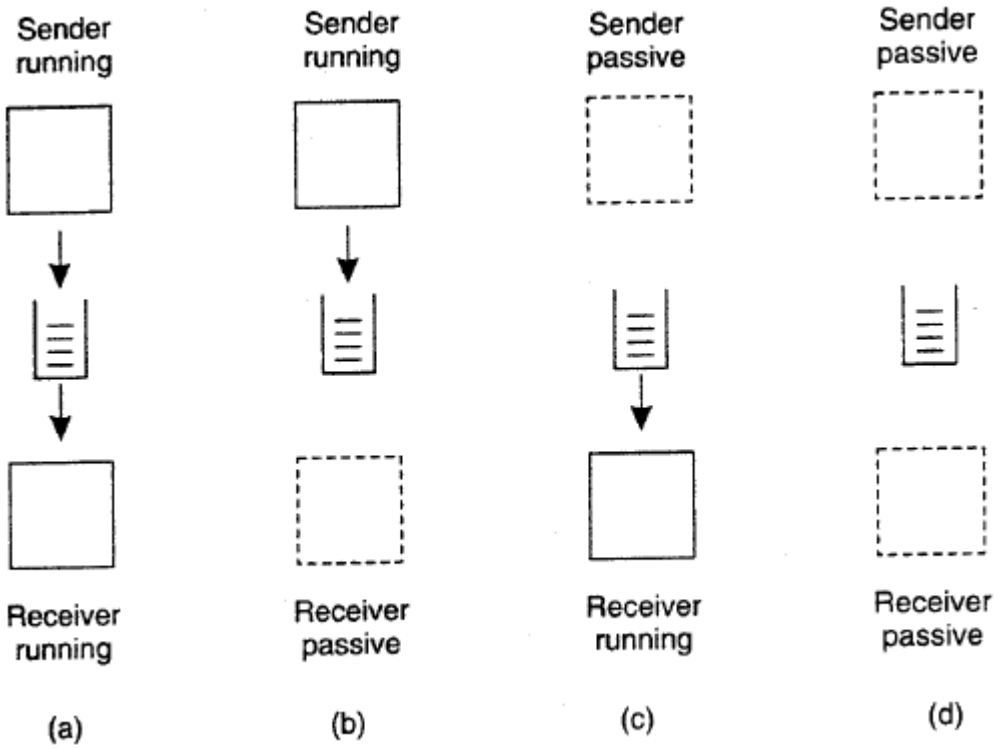


- Contoh : Message Passing Interface (MPI)

Message-oriented Persistent Communication

Message queuing model:

- Application communicate by inserting message in specific queues
- Forwarded over a series of communication servers
- Eventually delivered to destination



Message Queuing Model

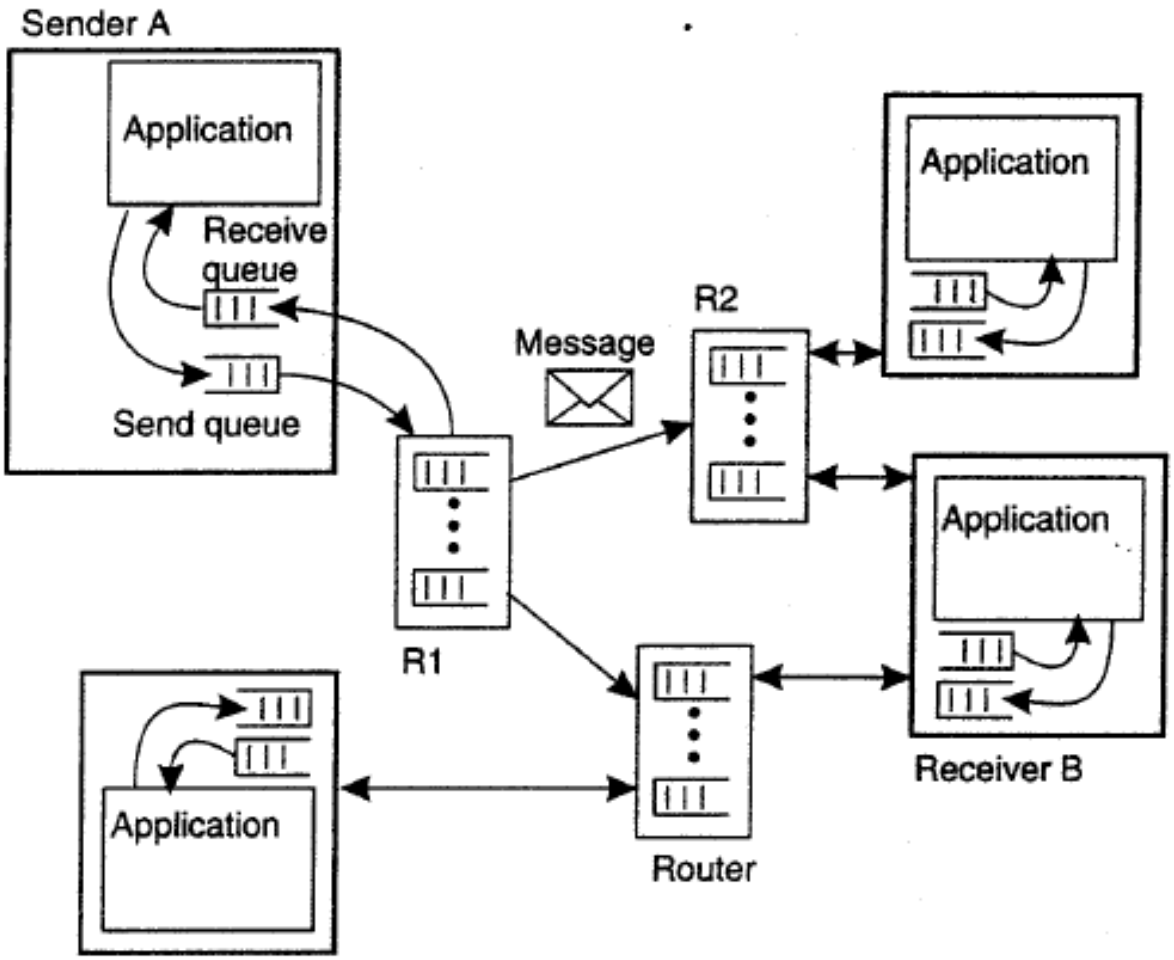


Figure 4-20. The general organization of a message-queuing system with routers.

Message Broker

Observation

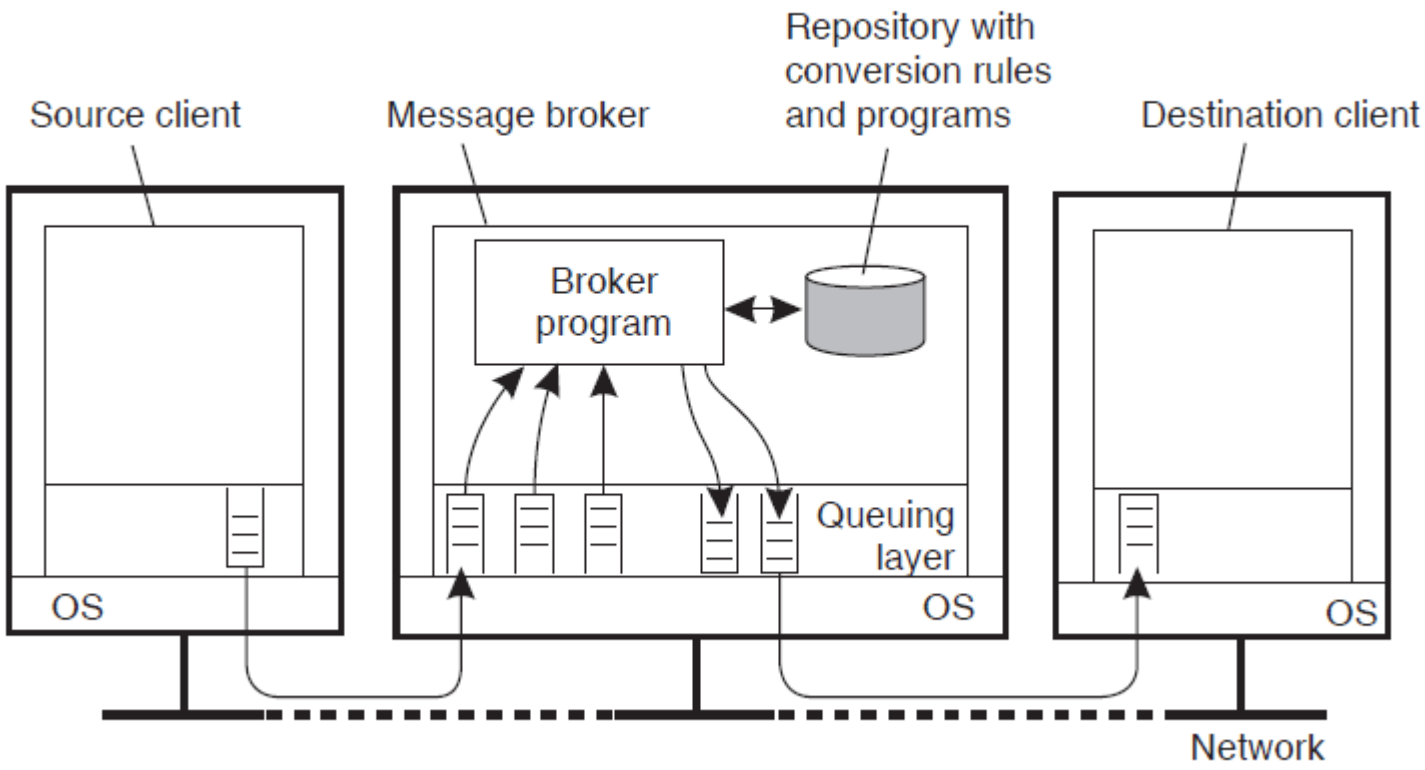
Message queuing systems assume a common messaging protocol: all applications agree on message format (i.e., structure and data representation)

Message broker

Centralized component that takes care of application heterogeneity in an MQ system:

- Transforms incoming messages to target format
- Very often acts as an application gateway
- May provide subject-based routing capabilities $\Rightarrow$ Enterprise Application Integration

Message Broker



Remote Procedure Call (RPC)

RPC

- Pemrograman client server menggunakan socket cukup sederhana
  - [connect]
  - Read/write
  - [disconnect]
- Namun tetap lebih mudah menggunakan abstraksi procedure/function call
- 1984: Birrell & Nelson mengajukan abstraksi Remote Procedure Call

Procedure call

- Misal:
  - x = f(a, “text”, 5);

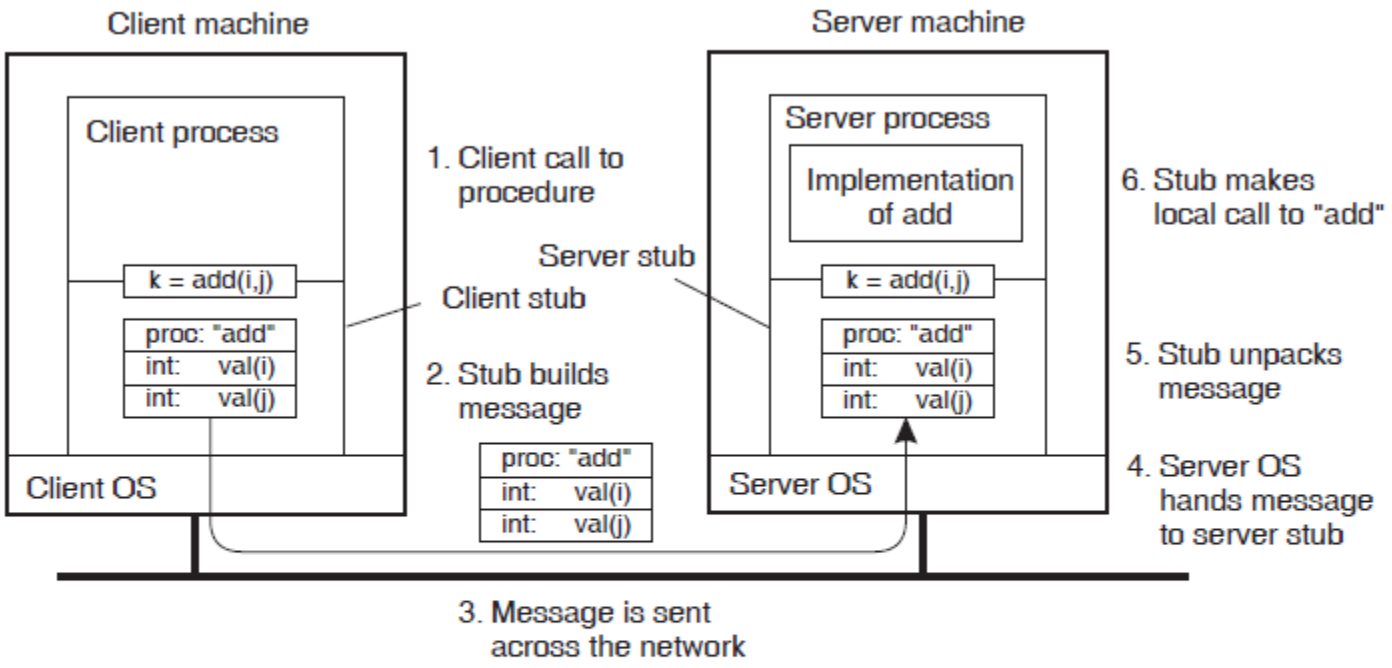
- Compiler akan mem-parse kode ini dan membangkitkan code untuk:
  - push nilai 5 ke stack
  - push address string “text” ke stack
  - Push isi variabel a ke stack
  - Memanggil/call fungsi f
  - Saat kembali dari stack, fungsi f akan menyimpan return value pada register

Remote Procedure Call

- Untuk menyediakan pemanggilan prosedur remote, digunakan mekanisme stub, sehingga pemanggilan prosedur remote dilakukan terhadap stub, dan stub yang menangani pengiriman pesan ke remote server

- Stub: prosedur/fungsi lokal yang memiliki interface sama dengan prosedur remote yang akan dipanggil

Basic RPC Operation



1. Client procedure calls client stub.
2. Stub builds message; calls local OS.
3. OS sends message to remote OS.
4. Remote OS gives message to stub.
5. Stub unpacks parameters and calls server.
6. Server returns result to stub.
7. Stub builds message; calls OS.
8. OS sends message to client's OS.
9. Client's OS gives message to stub.
10. Client stub unpacks result and returns to the client.

RPC

- RPC memberikan interface procedure call ke programmer
- Memudahkan pembuatan aplikasi
  - Kompleksitas kode jaringan disediakan oleh stub function
  - Programmer tidak perlu menangani detil seperti
    - Socket, port number, byte ordering
- RPC setara dengan presentation layer pada 7 layer OSI

Parameter passing

- Parameter marshaling: packing parameters into a message

- Pass by value
  - Data dikopi ke jaringan
  - Tidak masalah selama client dan server identical
  - Masalah representasi data

- Pass by reference
  - Tidak relevan jika tidak ada shared memory

Pass by reference

- Copy item yang di-reference ke message buffer
- Kirim via jaringan
- Unmarshal data di server
- Pass local pointer ke server stub
- Kirim nilai baru ke client

- Setiap struktur kompleks harus diubah menjadi representasi non pointer dan dikirim

Representasi data

- Pada pemanggilan lokal, tidak ada masalah tentang perbedaan format data

- Pada pemanggilan remote, mesin server dapat memiliki
  - Perbedaan byte ordering
  - Perbedaan ukuran integer dan tipe lainnya
  - perbedaan representasi floating point
  - Perbedaan character set
  - Perbedaan aturan alignment

Representasi data

- RPC membutuhkan standar encoding untuk komunikasi antar mesin
- Sun RPC menggunakan XDR (eXternal Data Representation)
- ISO menyediakan standar ASN.1 (Abstract Syntax Notation)
- Praktek umum saat ini: JSON, XML, Protocol Buffer, Apache Avro etc
- Bahasa tertentu (e.g. Java, Python) menyediakan format serialisasi

Representasi Data

- Saat dikirim, data dapat dikirim menggunakan
  - Implicit typing: hanya nilai yang dikirim, bukan tipe data atau parameter info, digunakan pada XDR
  - Explicit typing: type dikirim bersama nilai, digunakan pada ASN.1 dan XML, Protocol Buffer, Apache Avro

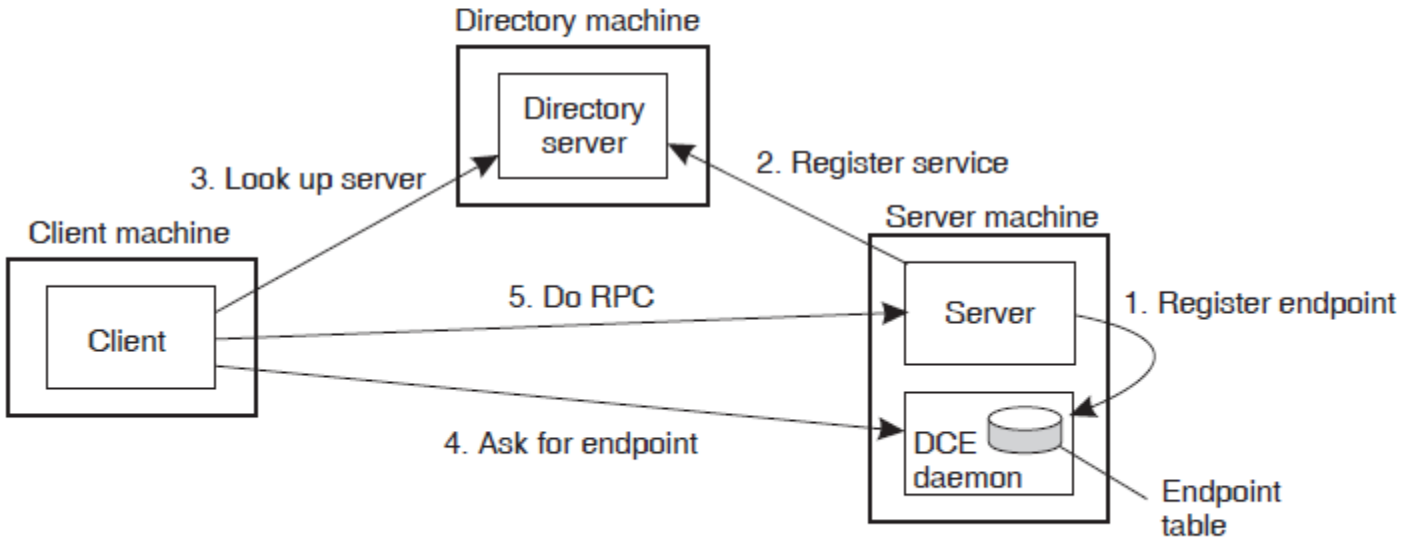
Binding

- Saat pemanggilan, client perlu menentukan host dan proses yang benar untuk sebuah RPC
- Birrell & Nelson (1984) mengajukan mekanisme database terpusat yang menyimpan informasi server dan proses mana yang menyediakan implementasi RPC tertentu
- Pada Sun RPC, database dikelola oleh masing2 host yang menyediakan RPC
  - Banyak host yang menyediakan layanan NFS, namun untuk file system yang berbeda2.

Client to server binding (DCE)

Issues

(1) Client must locate server machine, and (2) locate the server.



Transport protocol

- Beberapa implementasi RPC menggunakan hanya TCP
- Ada yang menyediakan beberapa transport protocol

Penanganan error

- Pemanggilan local call tidak gagal
  - Jika gagal, keseluruhan proses juga akan gagal
- Namun pada RPC, server dapat gagal independen terhadap client/pemanggil
- Aplikasi harus menangani kemungkinan2 kegagalan pemanggilan RPC

- Semantik RPC
  - Pada local call, semantiknya adalah exactly once
- Pada RPC, remote procedure call dapat dipanggil
  - 0 kali: jika server crash sebelum menjalankan kode server
  - 1 kali: jika semua berjalan baik
  - 1 or more: jika terjadi delay atau lost reply, sehingga client mengirim ulang pesan

Semantik RPC

- Umumnya, implementasi RPC menyediakan semantik:
  - At least once
  - At most once
- Perlu memahami karakteristik aplikasi
  - Idempotent: fungsi dapat dipanggil berulangkali tanpa side effect
  - Non idempotent: fungsi tidak boleh dipanggil berulangkali

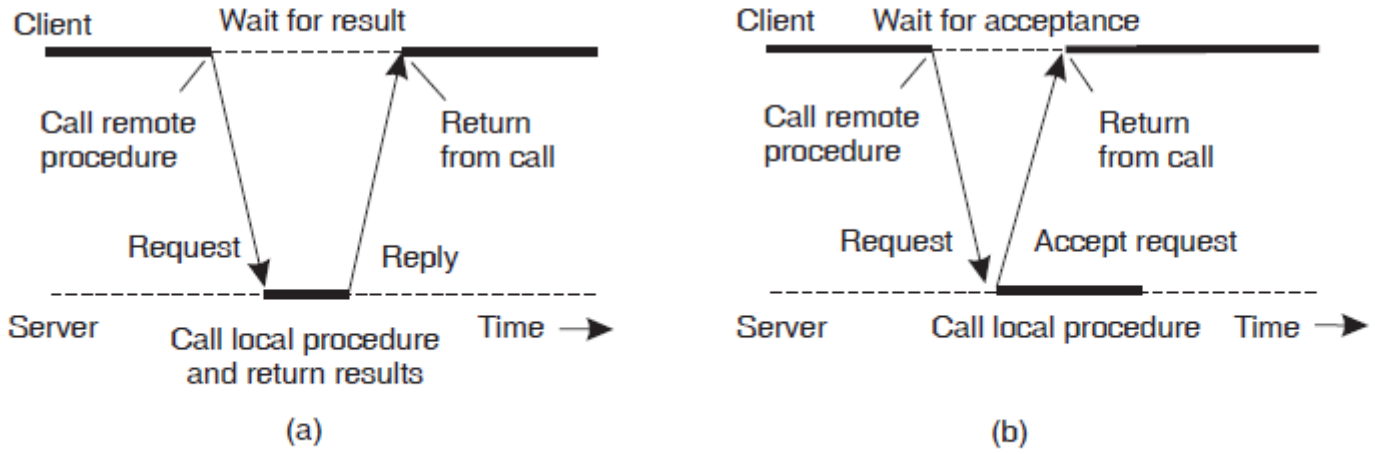
Isu lain

- Kinerja
  - RPC jauh lebih lambat dibandingkan local call
- Security
  - Message visible over network
  - Authenticate client
  - Authenticate server

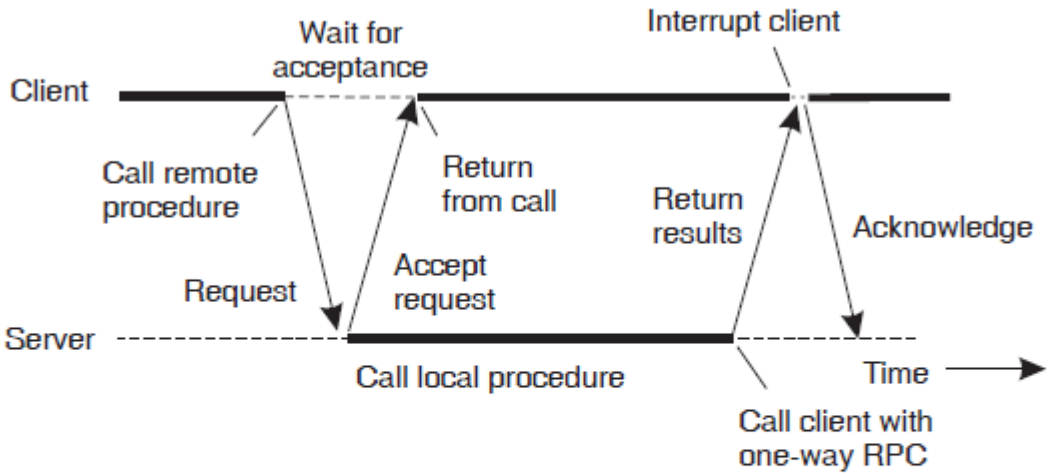
Asynchronous RPC

Essence

Try to get rid of the strict request-reply behavior, but let the client continue without waiting for an answer from the server.



Deferred Synchronous RPC



Variation

Client can also do a (non)blocking poll at the server to see whether results are available.

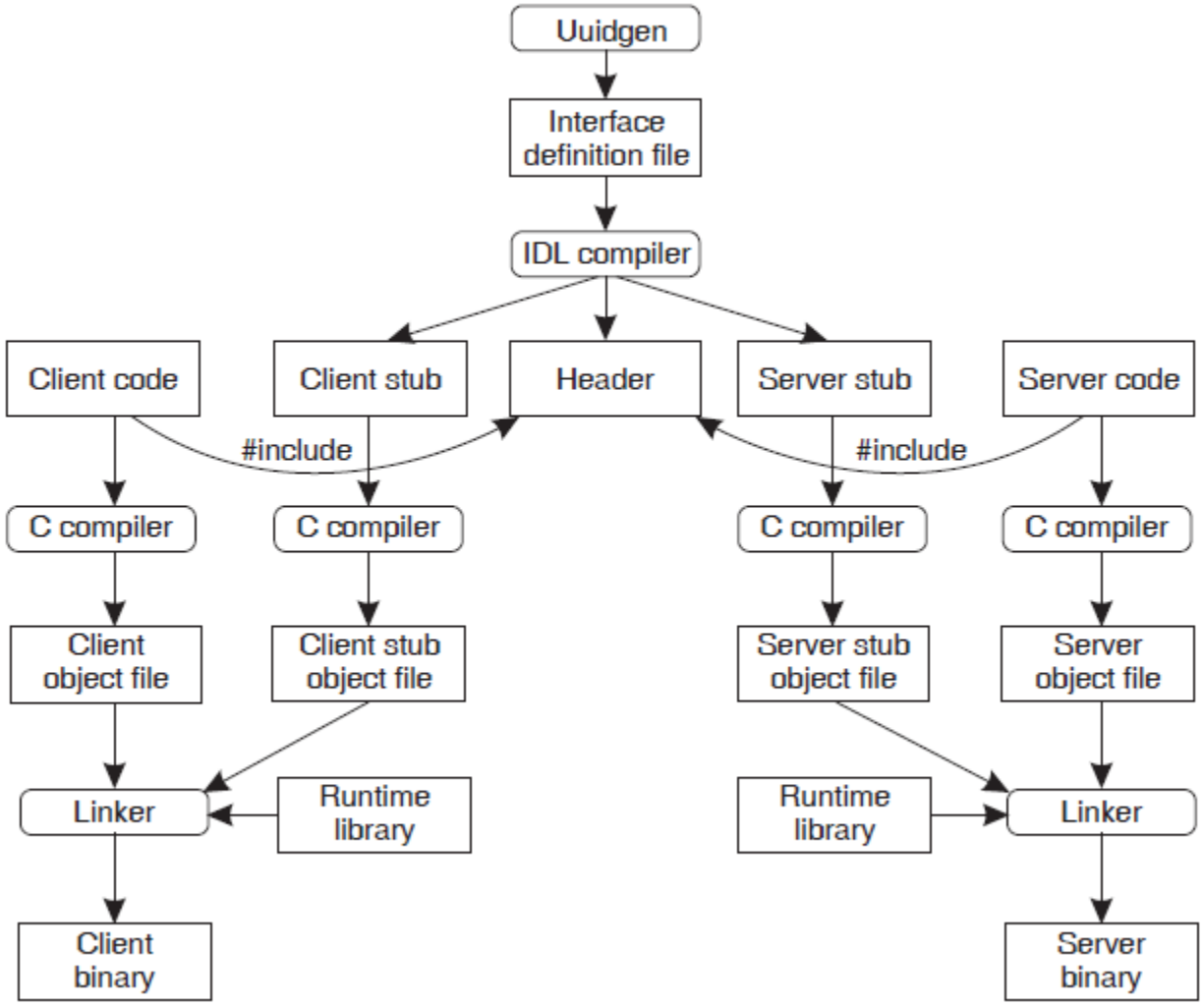
Pemrograman dengan RPC

- Dukungan bahasa
  - Kebanyakan bahasa pemrograman tidak menyediakan dukungan konsep RPC
  - Digunakan kompiler terpisah untuk membangkitkan kode stub

Interface Definition Language

- Memungkinkan programmer untuk menspesifikasikan interface prosedur
- Pre-compiler menggunakan spesifikasi ini untuk membangkitkan kode stub client dan server
  - Marshaling code
  - Unmarshaling code
  - Network transport routines
  - Conformed to defined interface

RPC in practice



ONC (Sun) RPC

- ONC: Open Network Computing
- Dikembangkan oleh Sun (sekarang Oracle)
- RFC 1831 (1995), RFC 5531 (2009)
- Tetap digunakan karena dipakai pada NFS (Network File System)
- Interfaces didefinisikan dengan Interface Definition Language (IDL)
- • IDL compiler: *rpcgen*

Stream Oriented Communication

Peran Timing dalam komunikasi

- RPC maupun MOM melakukan exchange data dalam bentuk independent dan complete units of informations
- Tidak ada masalah terhadap kapan persisnya komunikasi terjadi
  - Timing has no effect on correctness
- Terdapat model komunikasi lainnya dimana timing merupakan hal krusial

Data Streaming

- Diperlukan model komunikasi untuk exchange time-dependent information seperti audio dan video
- Pengolahan data stream kontinu, seperti event log, IoT sensor
- Unbounded data, data mengalir terus

- A data stream is nothing but a sequence of data units

- Simple stream
  - terdiri atas 1 sequence of data
- Complex Stream
  - Terdiri atas beberapa simple stream yang berelasi (substream)
  - Relasi antar substream bersifat time dependent
  - Contoh : streaming film yg terdiri atas substream video & substream audio

- Push model
  - Produksi data dikendalikan oleh sumber
  - Publish/subscribe model

- Konsep waktu
  - Kadang diperlukan untuk menentukan kapan data diproduksi dan kapan output dihasilkan
  - Time agnostik, processing time, ingestion time, event time

Stock market
- Impact of weather on securities prices
- Analyze market data at ultra-low latencies

Natural systems
- Wildfire management
- Water management

Transportation
- Intelligent traffic management

Manufacturing
- Process control for microchip fabrication

Health and life sciences
- Neonatal ICU monitoring
- Epidemic early warning system
- Remote healthcare monitoring

Law enforcement, defense and cyber security
- Real-time multimodal surveillance
- Situational awareness
- Cyber security detection

Fraud prevention
- Multi-party fraud detection
- Real-time fraud prevention

e-Science
- Space weather prediction
- Detection of transient events
- Synchrotron atomic research

Other
- Smart Grid
- Text Analysis
- Who's Talking to Whom?
- ERP for Commodities
- FPGA Acceleration

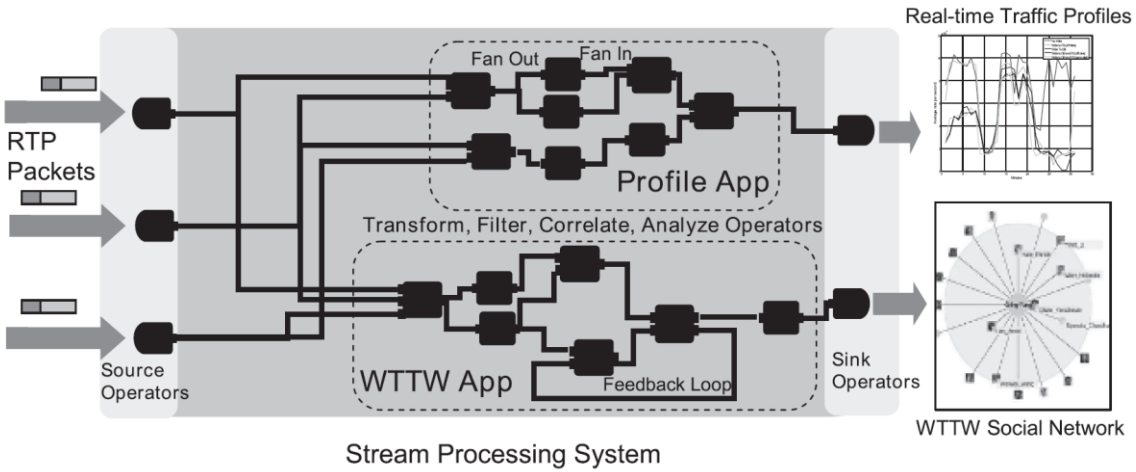
Telephony
- CDR processing
- Social analysis
- Churn prediction
- Geomapping

Figure 2.1 Stream processing applications.

Andrade et.al., Fundamentals of Stream Processing: Application Design, Systems, and Analytics, 2014

Stream Processing

- Didefinisikan berupa information/data flow
- Data source
- Processing
- Data sink



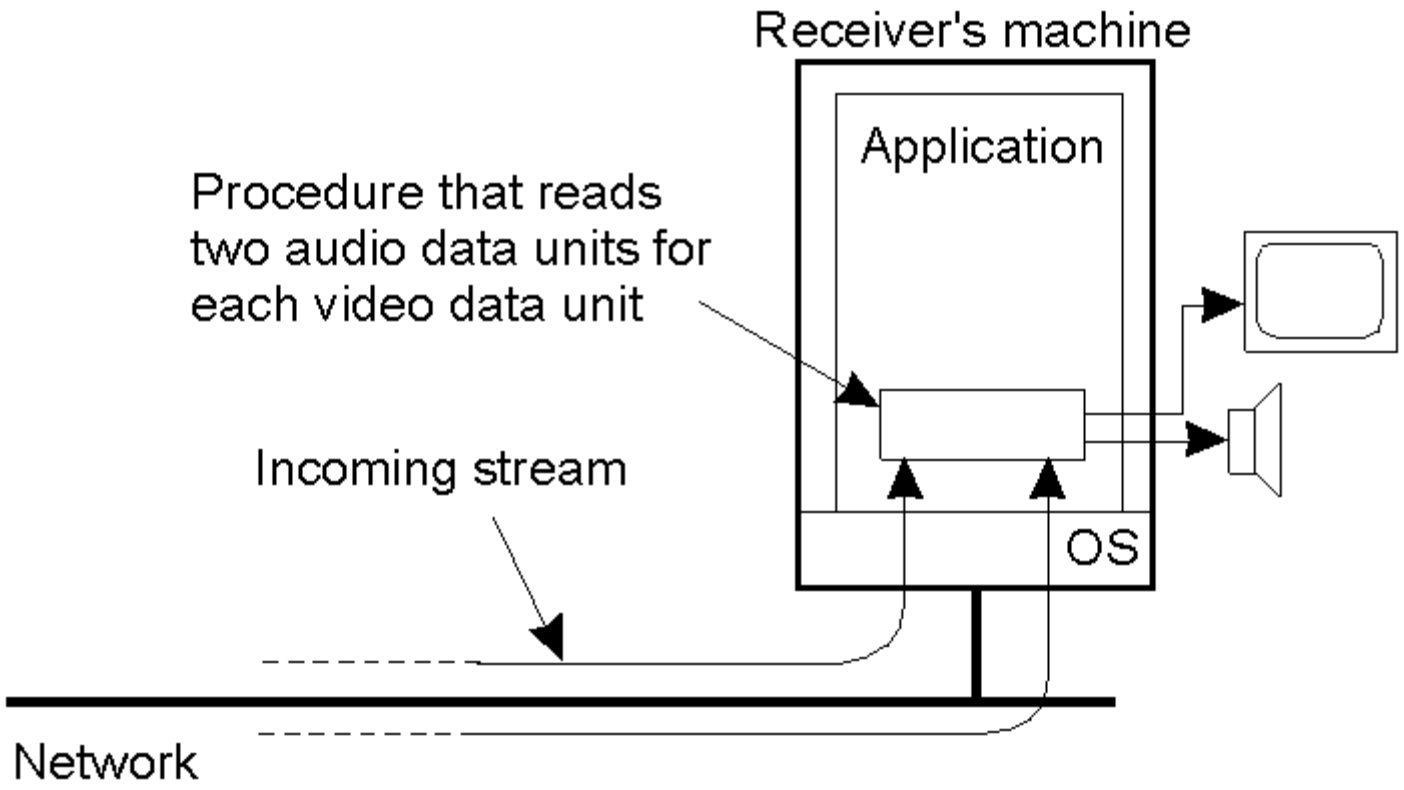
Quality Of Service (QoS)

- Requirement terhadap service pengiriman data dituliskan dalam bentuk QoS
  - Termasuk timing
- QoS untuk continuous data stream umumnya berkaitan dengan timeliness, volume, dan reliability
- Contoh kakas yang dikembangkan untuk continuous data stream: Apache Kafka, Apache Storm, Apache Hadoop, Apache Spark, Apache Flink, Apache Samza

Contoh QoS

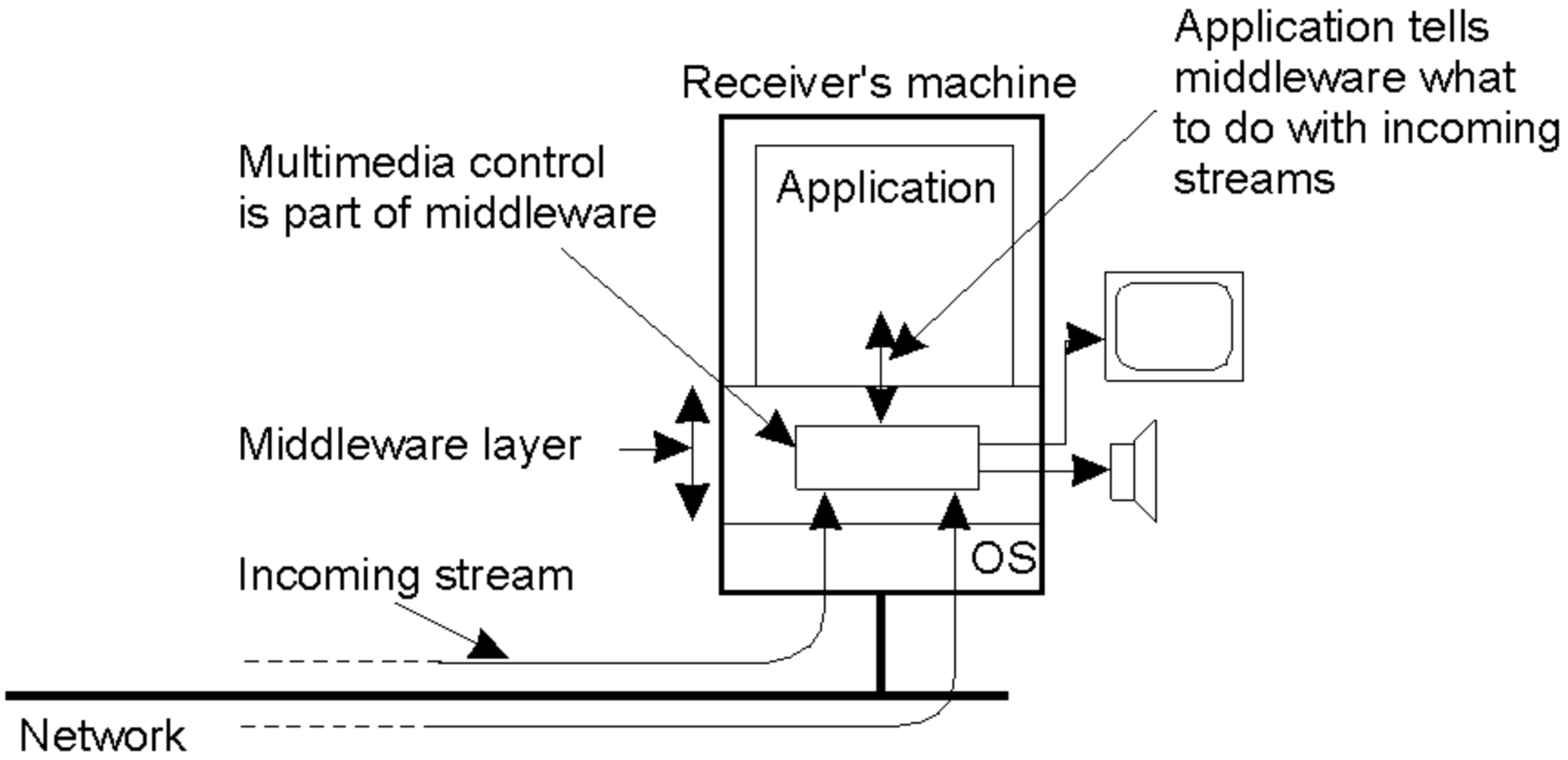
- Berikut contoh QoS dari perspektif aplikasi
  - Required bit rate which data should be transported
  - Maximum delay until a session has been set up
  - Maximum end-to-end delay
  - Maximum delay variance (jitter)
  - Maximum round trip delay

Stream Synchronization (1)



- The principle of explicit synchronization on the level data units.

Stream Synchronization (2)



- The principle of synchronization as supported by high-level interfaces.

Referensi

- Andrew S. Tanenbaum and Maarten Van Steen. Distributed System Principles and Paragdims. 2007