Model of Middleware

Model of middleware

- **Distinct set of services required**
  - **to support distributed processing and**
  - **to support heterogeneous computing**

  → To support for web service in ubiquitous environmnet

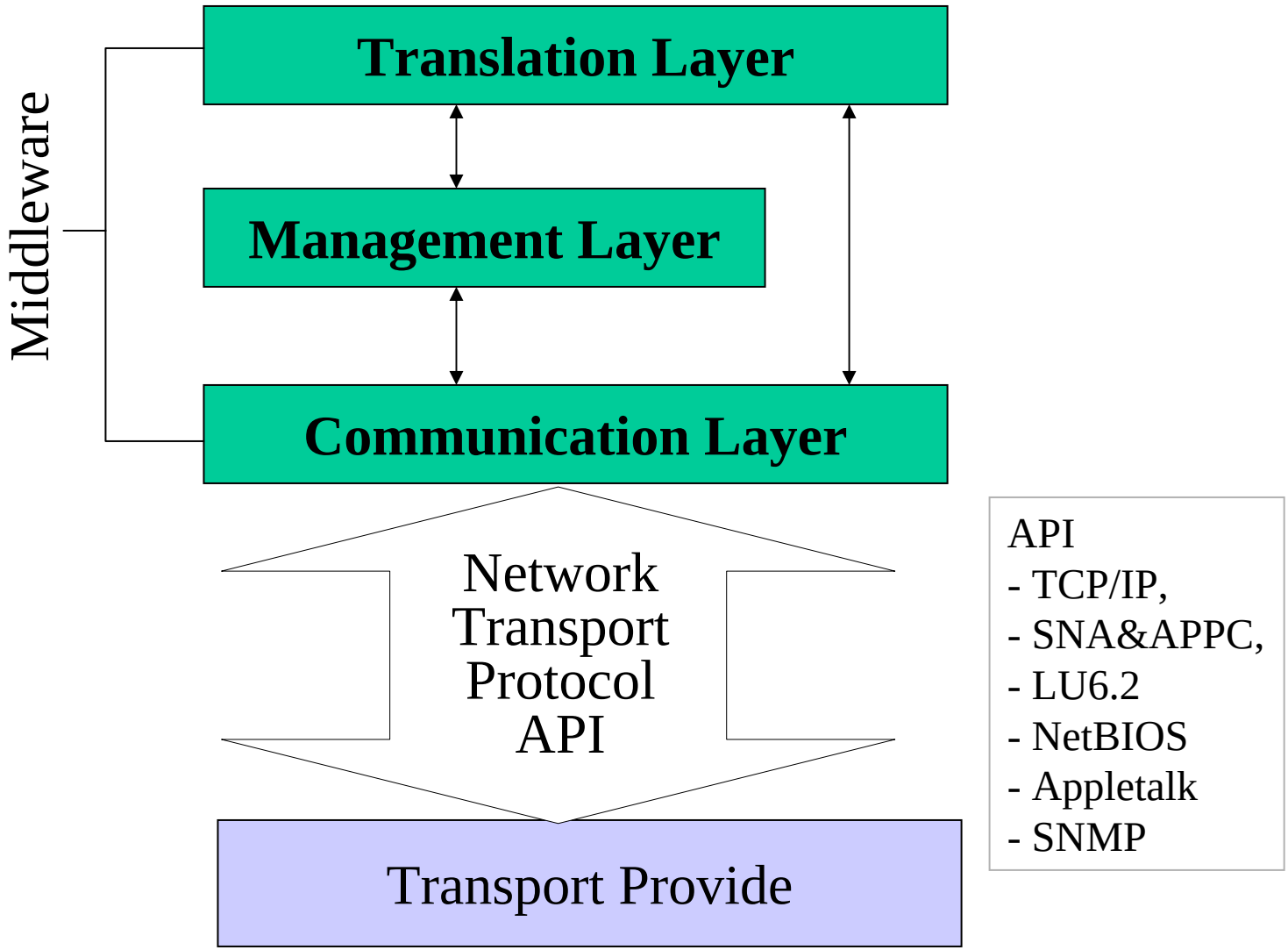
    - **Translation Layer**
      - **DBC, connectivity**
      - **Enhanced for Multimedia Contents**
    - **Communication Layer**
      - **RPCs, MOMs**
      - **Enhanced for Web**
    - **Management Layer**
      - **DCE, DTPM, ORB**
      - **Enhanced for Web Service**

Domain Specific MW

- Control Middleware
  - Applicance, Context-Aware, Power-Aware
  - Ex : Home Netowork, Sensor Network
- Multimedia Middleware
  - Stream service
  - Ex : IPTV, DMB
- Distributed Object Middleware
  - Corba, DCOM
- Home Network Middleware
  - HAVi, UPnP, LonWorks, JINI
- Web Service Middleware
  - Semantic Middleware

- All kinds of Doman-Depend Middleware

Model of middleware



Translation Layer (1)

- What it does
  - enables interoperability between heterogeneous SW products
  - increases the adaptability of applications by making target products
  - makes heterogeneous environments less complex by reducing the number of APIs developers have to work with

  $\Rightarrow$ achieve this by providing a common API
    - ODBC API: translates the ODBC calls into the native calls of the target DBMS

  $\Rightarrow$ offer a ‘fix’ to specific interoperability problems
    - translates calls to one API into the format required by another API
    - MS’s windows API can be translated into the Motif API

Translation Layer (2)

- The limits of translation
  - how to handle the inevitable mis-match of functionality between products
    - no one-to-one correspondence between the functionality of one product and that of another

  $\Rightarrow$ Solving approaches:
    - common factor
    - superset
    - sub-set
- Product-to-product translation
  - to support inter-product communication between any products

    ex) - mail product $\rightarrow$ use middleware to access DBMSs/OSs

    - DBMSs $\rightarrow$ use middleware to access OS

    - word-processor SW $\rightarrow$ use middleware to windowing SW

Translation Layer (3)

- Common factor
  - simplest approach by using an API
    - encapsulates the concepts and functions shared by all the products
    - provides a ‘pass-through’ by middleware suppliers
      - native API calls are made unaltered
      - access to product-specific functionality but sacrifices portability and openness
    - negates the poratability, flexibility, and a single API for whole range of products

$\Rightarrow$ Because of these limitation, rarely used

Translation Layer (4)

- Superset approach
  - provide an API to encapsulate all the commands and all the concepts of the products
    - must emulate those functions that cannot themselves support

$\Rightarrow$ Beyond the capabilities and budgets of the middleware suppliers

- extremely time consuming
- no longer cost effective

Translation Layer (5)

- Sub-set approach
  - an approach between the common factor and the superset approach
    - chooses the functions in the combined products
    - provides an API to handle these functions
  - two options for the supporting
    - provide **an emulation capability** for those functions alone
    - provide **a form of pass-through** which uses the supplier's API to access the unique functionality of some products

$\Rightarrow$ **affect the portability and openness** because the use of functions are not available in all products limits adaptability

- developers: has to know the functionality of product
- suppliers: package the APIs for a specific product into modules

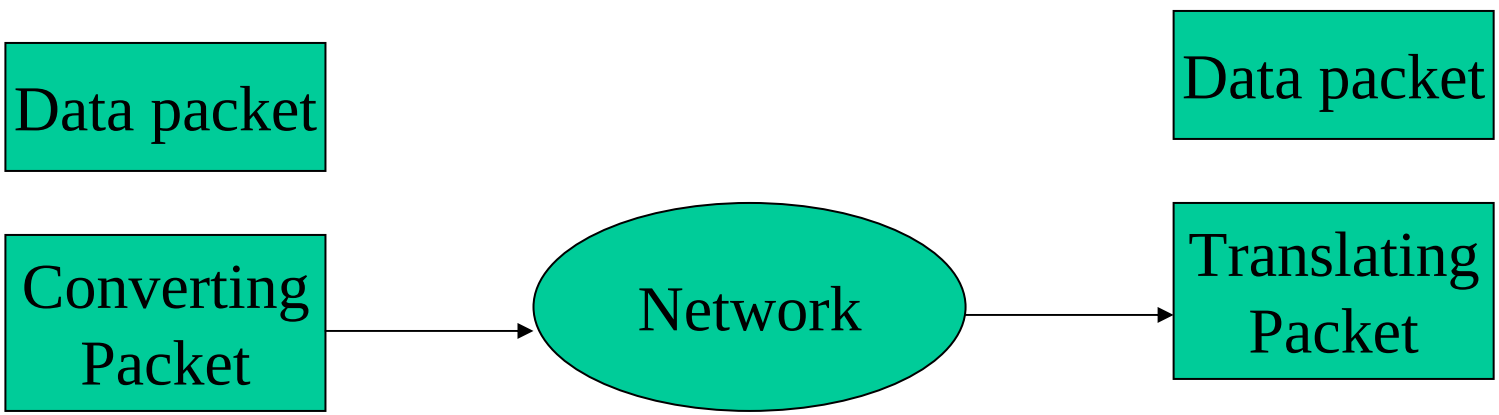
Communication Layer (1)

- What does it
  - enables a developer **to ignore** the fact of networking
  - provides **a 'virtual machine'** to developers
    - hides the complexity of a distributed, heterogeneous environment
  - uses the network SW to transport instructions and data across the network

  $\Rightarrow$ Responsible for:
    - converting data formats
    - packaging instructions and data for transport
    - providing address information used by the network SW
    - conforming with the network protocol
    - establishing and maintaining communication between sending and receiving processes
    - capturing error messages returned by the network SW

Communication Layer (2)

- Converting data formats
  - converts different data formats for networking
    - be done using an intermediate format
    - converts the data into the device-independent format
    - translates into the format expected by the receiving machine



Communication Layer (3)

- Packaging instructions and data
  - packages the instruction and data to be passed via network
  - adds other information for transmission control
    - data compression, checksums, or encryption
  - unpacked at the receiving time

- Providing addressing information
  - be responsible for providing the addressing information to transport to the appropriate location
    - location transparency
  - ensures that any communication is correctly addressed to the required location

$\Rightarrow$ the task of **middleware**

Communication Layer (4)

- Conformance with network protocols
  - network protocol: defines
    - how computers should communicate over a network
    - how computers should identify one another on a network
    - the data should take in transit
    - how the information should be processed once it reaches its final destination
  - provides a high-level interface to a specific low-level protocol
    - convert the high-level to the lower-level statements
  - provides a common interface to the many protocols
    - ex: Berkeley Sockets

$\Rightarrow$ a proprietary API developed by the vendor of the **middleware**

- faced with the same problem whether to take a common factor, subset, super-set

Communication Layer (5)

- Establishing and maintaining communications between processes
  - handles the communication between processes
    - communication session between machines on the network
  - range of inter-process communication styles
    - synchronous
    - asynchronous
    - session-less
    - message-based
    - using a tightly-coupled dialogue style
    - using strored-end-forward methods involving queues

$\Rightarrow$ one of the key differences between **middleware products**

Communication Layer (6)

- Capturing errors
  - detects an error in the transport processes
  - returns the error to the initiator of the communication

  $\Rightarrow$ supports error handling and error recovery
    - is part of management layer

- Network gateway
  - enable one network product to talk to another network product
    - by translaton between network protocols
    - embedded in hardware products

  $\Rightarrow$ will be **transparent to any middleware**

Management Layer (1)

- What it does
  - offers a natural extension of the basic services provided by communication layer
    - controls the message corruption, loss, and error
    - checks the message loss and handles the abnormal situation
      - » the responsibility of providing these services from the developers

Management Layer (2)

- Typical services
  - directory service:
    - directory management without affecting the processes by changing the locations
  - security service
    - verify the identity and authority level of ‘user’ of the network
  - guaranteed delivery service
    - ensure message and calls reach their destination
      - extend to full transaction recovery across the network
  - exception handling, fault and problem management service
    - provides a recovery co-ordination capability

Management Layer (3)

- Typical services (Continue)
  - time service
    - a means of agreeing time between several hosts and program
    - mainly used by recovery and auditing functions
  - synchronization service
    - handle parallelism,
    - asynchronous requests: request with no response
    - synchronous requests: requests that wait for a reply
    - deferred synchronous requests: processes that carry on after sending the request and claim the reply later

Management Layer (4)

- Typical services (Continue)
  - thread service
    - supports the multi-threading
  - multi-tasking service
    - support the multi-tasking
  - compression service
    - provides compression and decompression of packets or messages

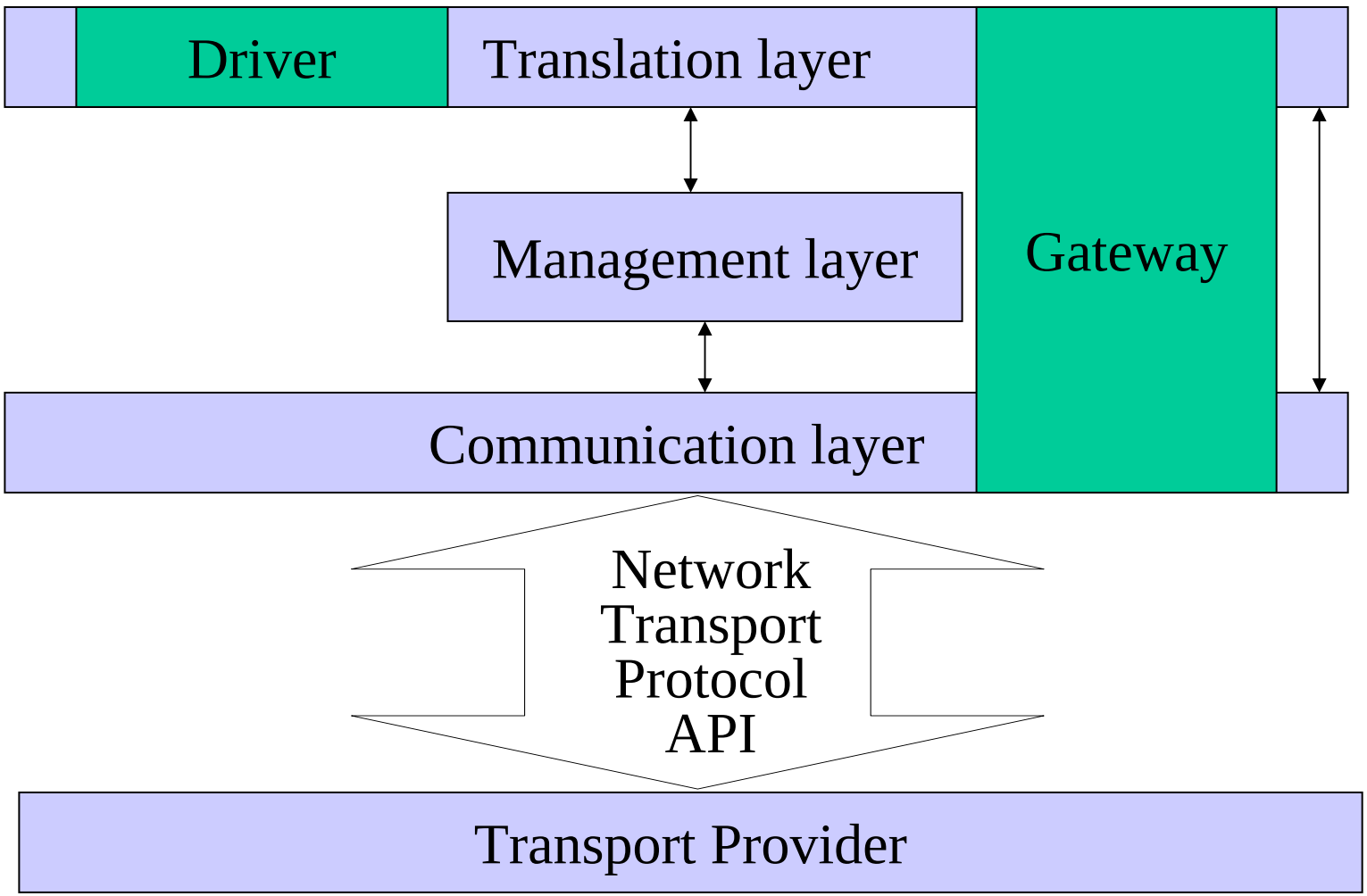
Management Layer (5)

- Additional services
  - capacity planning
    - collects data on network traffic loads and server usage
    - assist with performance monitoring and capacity planning
  - load balancing
    - responds dynamically to traffic patterns when overload of network resources is detected
      - resources: nodes, links, processes, and servers
      - by re-routing messages, providing feedback to the sending process and adjusting loads
      - used information on message priorities and traffic data

Products for the translation layer (1)

- Products for the translation layer
  - enable heterogeneous SW to inter-operate
  - ex: database connectivity products
    - translate between the different DML of DBMSs
    - provide a common API to access different DBMSs
- Drivers and Gateways
  - drivers: only support the translation
    - » no management and communication services
  - gateways: support both the translation and the communication
    - » translates APIs and
    - » enables transport of the instructions to remote locations over network
    - » ex: RPC, Messaging middleware, or low-level communication service products
    - see Figure in Next Slide

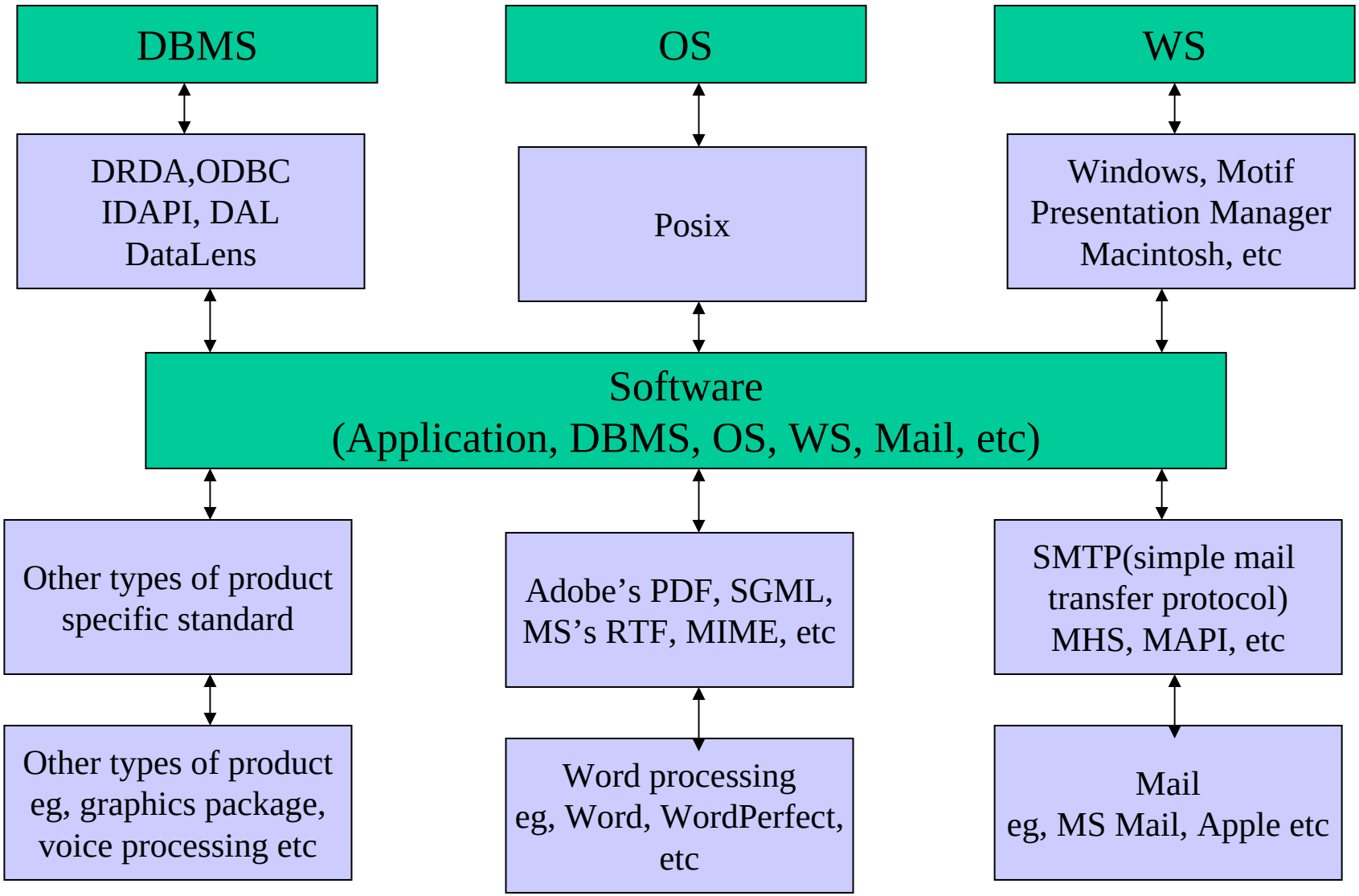
Driver and Gateway



Products for the translation layer (2)

- Forms of translation
  - Product types for requiring the translation services
    - DBMSs: Ingres, Oracle, Informix, Sybass, Access, etc
    - Operating Systems: MVS, VME, HP-UX, SunOS, Solaris
    - Windows Systems: Windows, OS/2, etc
    - Word Processors: see Chapter J
    - Spreadsheets: see Chapter J
    - Drawing Packages : see Chapter J
    - Mail and Fax Packages : see Chapter J
  - See Figure in Next Slide

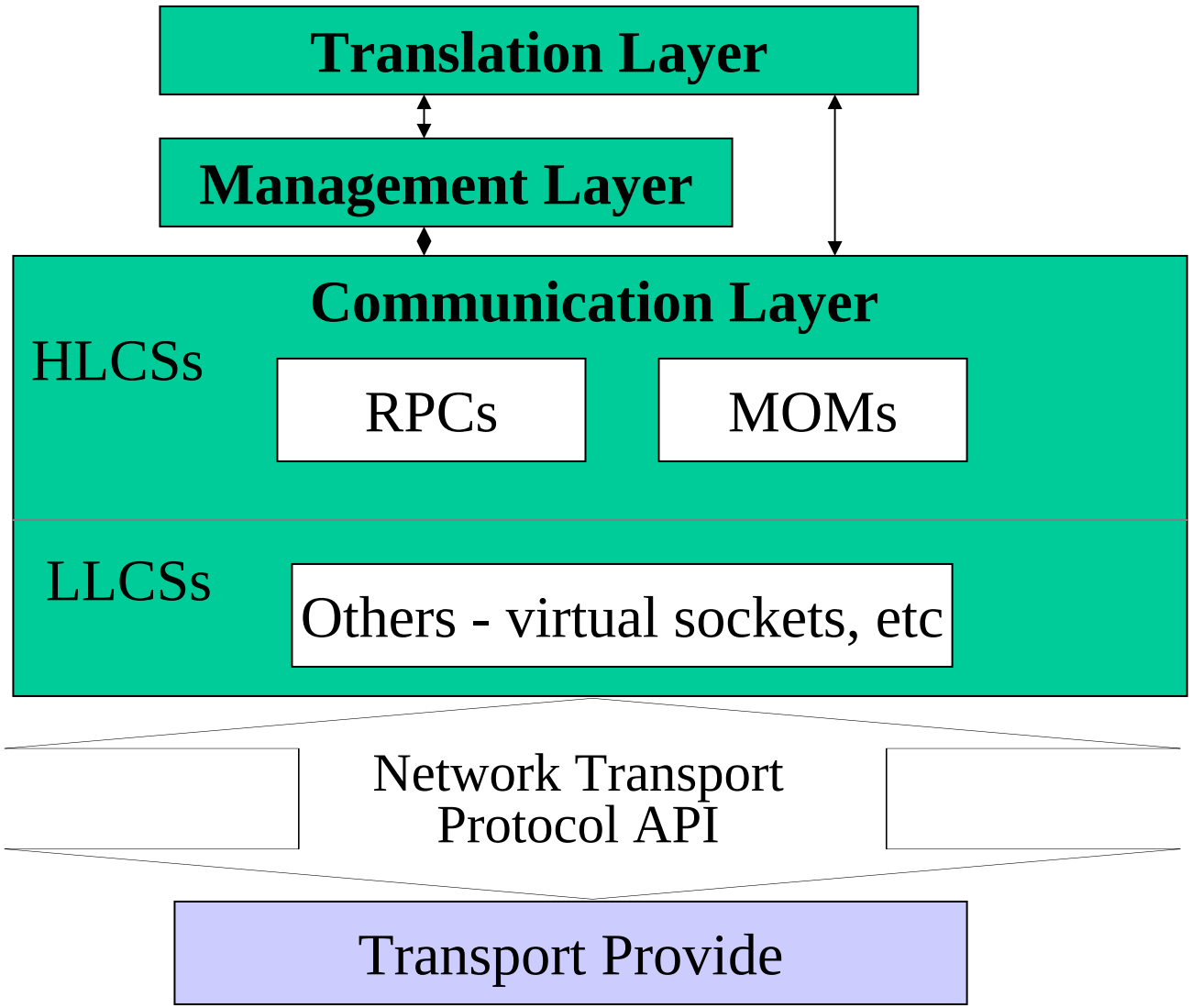
The Translation Layer in Detail



Products for the communication layer (1)

- Objectives
  - provide varying degrees of protection from network
    - occurs at the highest level of abstraction in mddleware
    - allows the developers to ignore the fact that communication takes place over a network
  - *high-level communication service* (HLCS) middleware
    - be used to write a distributed program by developers
    - makes the remote program appear to be local
  - low-level communication service (LLCS) middleware
    - provides a common API to access multiple network protocols and the protection at the network level

A Model for RPC/MOM



Products for the communication layer (2)

- Remote Procedure Calls
  - to hide the complexity of LLCS programming
    - use an interface for defining IPC
    - mimics the structure of a C function call
      - » a function's name
      - » its arguments when called
      - » its return value or structure
  - essentially point-to-point communication between one program and another
    - recently, provides a one-to-many communication capability
  - RPC tools
    - generate the layer of communication code needed

Products for the communication layer (3)

- Message-Oriented Middleware
  - an alternative approach to RPCs
    - uses the concept of a message for network independent communication
  - send messages containing data or instructions to other processes
  - acts as a form of reliable store-and-forward system
  - four stages for data moving
    - SP sends a message to MW
    - MW places the message onto a queue
    - MW sends the message and places it on the RP’s queue
    - RP reads the message from the queue

Products for the Management layer (1)

- Provides management services integrated with communication services
  - cannot exist in its own right

- Three main categories
  - Object request brokers: ORBs
    - Chapter G
  - Distributed computing environment: DCEs
    - Chapter H
  - Distributed/On-line transaction processing monitors: DTPM or OLTPM
    - Chapter I

Products for the Management layer (2)

- ORBs
  - communication between objects using message
  - handle request and responses, synchronization, delivery of the response, exceptions, and security
  - based on LLCS
  - using C++, Smalltalk, or other OOPL
- DCEs
  - provide an environment to support the directory, naming, and security services
  - based on RPC
- TPMs
  - manage the transactions across distributed, heterogeneous environment
  - provide recovery by using two-phase commit

Function creep between Middleware types