

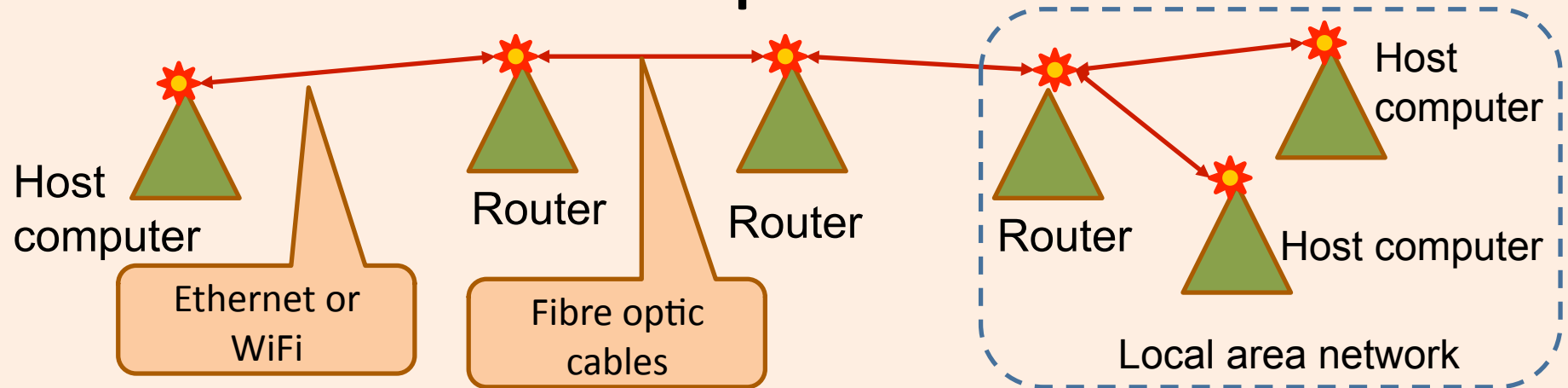
# ICS 321 Data Storage & Retrieval

## SQL in a Server Environment (i)

Asst. Prof. Lipyeow Lim  
Information & Computer Science Department  
University of Hawaii at Manoa

# Networking Primer

# Modern Computer Networks



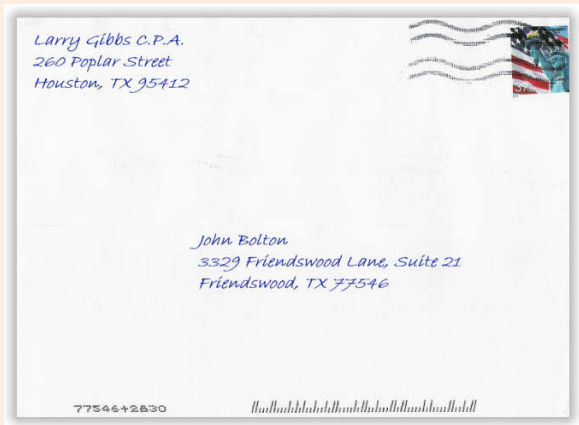
- Signaling technology can transmit complex sequences of bits - **packets**
- Each host or router obeys a set of rules for how to handle incoming/outgoing messages – communication **protocols**
- Communications can be multi-way
- **Bandwidth**: the number of bits that can be transferred per second (bps)
- **Latency**: the time it takes for a message to reach the destination after leaving the source

# Local Area Networks

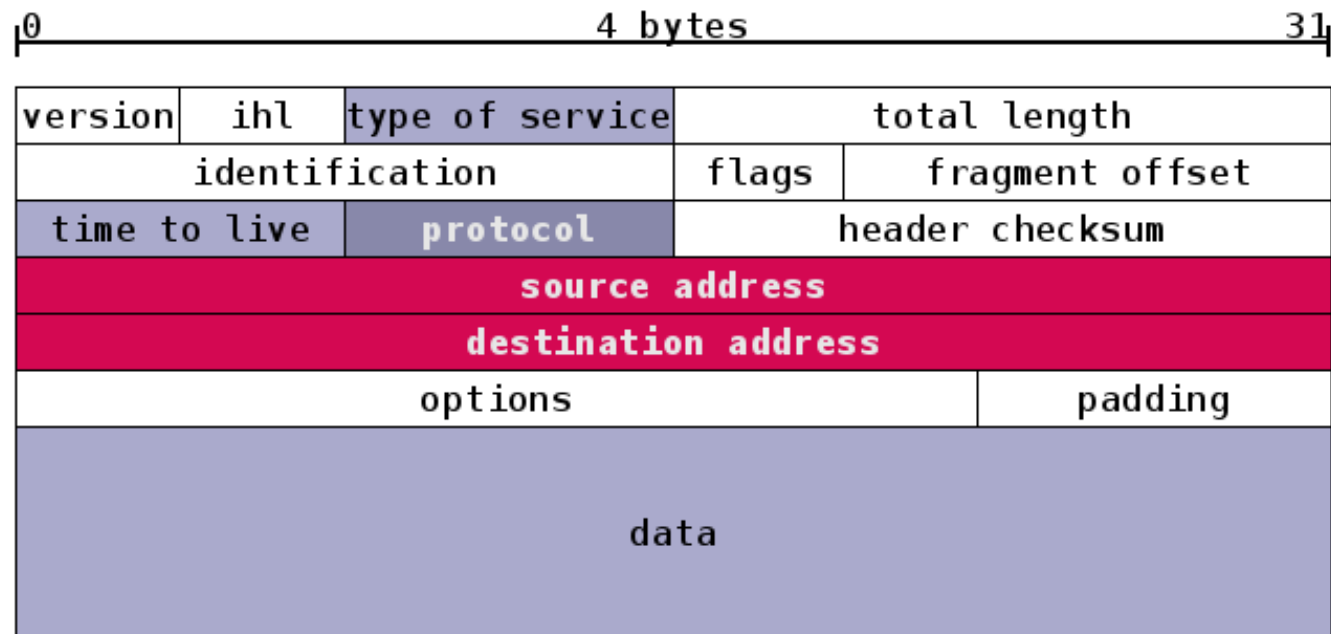


- Wired (UTP Cat5) or Wireless 802.11
- Connects hosts within a limited spatial region together to form a network
- All hosts within the network can “talk” to each other
- The network is often a shared medium: only one host can talk at one time and the rest listens.

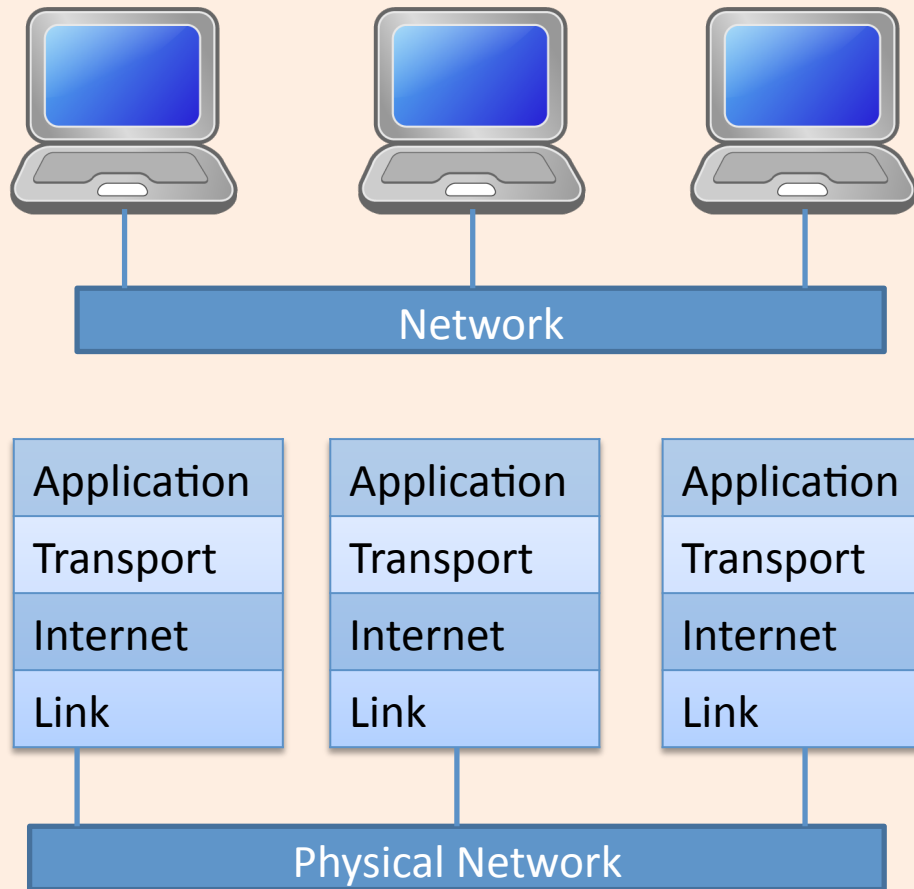
# Data Packet



- How messages are packaged for delivery on the network – like postal mail.
- Source and destination addresses

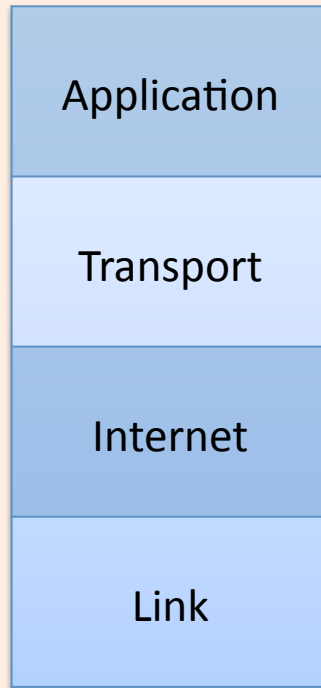


# Network Abstractions



- Network communications are conceived as layers of abstractions.
- Each layer plays a specific role and is relatively independent of other layers
- Each layer has its own packet format
- Packets from higher layers are embedded in packets of lower layers – “encapsulation”

# TCP/IP Four Layer Model



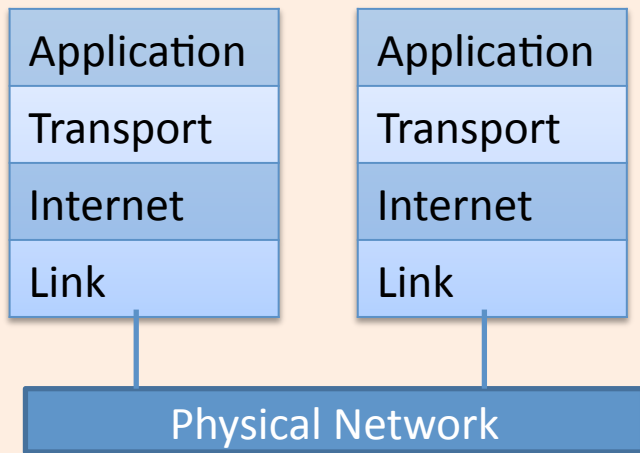
- Process to process: communicates data to other processes/applications on the same host or on other hosts
- Eg. SMTP, FTP, SSH, HTTP

- Host to host: communicates data to other host on the same network or on other networks
- Hides the topology of the network
- Flow control, error correction, connection control
- Eg. TCP, UDP

- Inter-network: communicates data to other networks
- Deals with addressing and routing of datagrams to next network
- Eg. IPv4, IPv6

- Transmit data to other network interfaces on the local network
- Eg. Ethernet, WiFi 802.11

# Link Layer



- Eg. Ethernet, WiFi 802.11
- A host can have multiple network interface cards (eg. Laptops typically have an ethernet interface and a WiFi interface)
- Each interface has a 48-bit physical address that is hardwired to the hardware

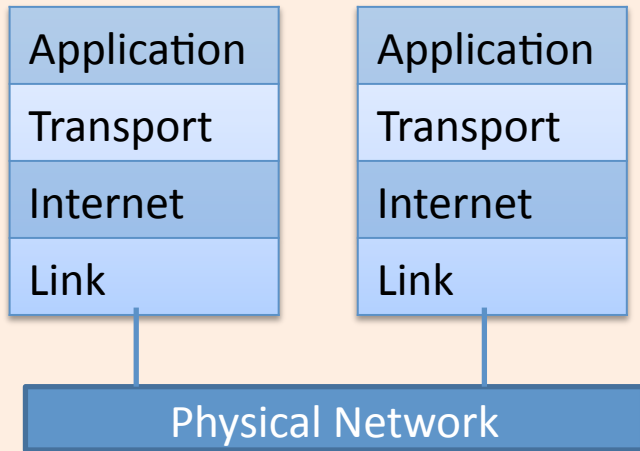
Data packet arrives from upper layer (Internet layer)

- If packet is too big, break packet into smaller fragments ('frames')
- Embed data packet in a link layer packet with link layer header, sequence number, error correction code etc.
- Link layer packets gets transmitted on physical link
- Link layer protocol governs how transmission over physical link is done. Eg. Carrier sense multiple access

Bottom-up process is similar on the receiving host



# Internet Layer



- Eg. IPv4
- Connects multiple networks together.
- Each network interface of a host is associated with an 32-bit IPv4 address
- IP address is not hardwired, but assigned in the software

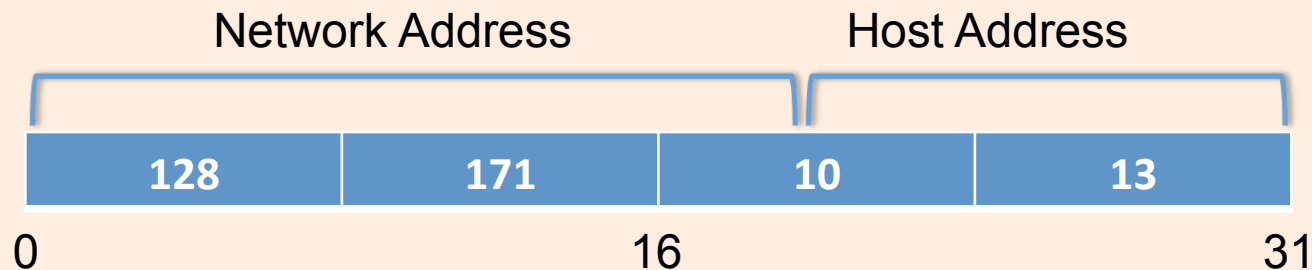
Data packet arrives from Transport layer

- Embed data packet in an IPv4 packet with IP header etc.
- Pass packet to Link layer

Data packet arrives from Link layer

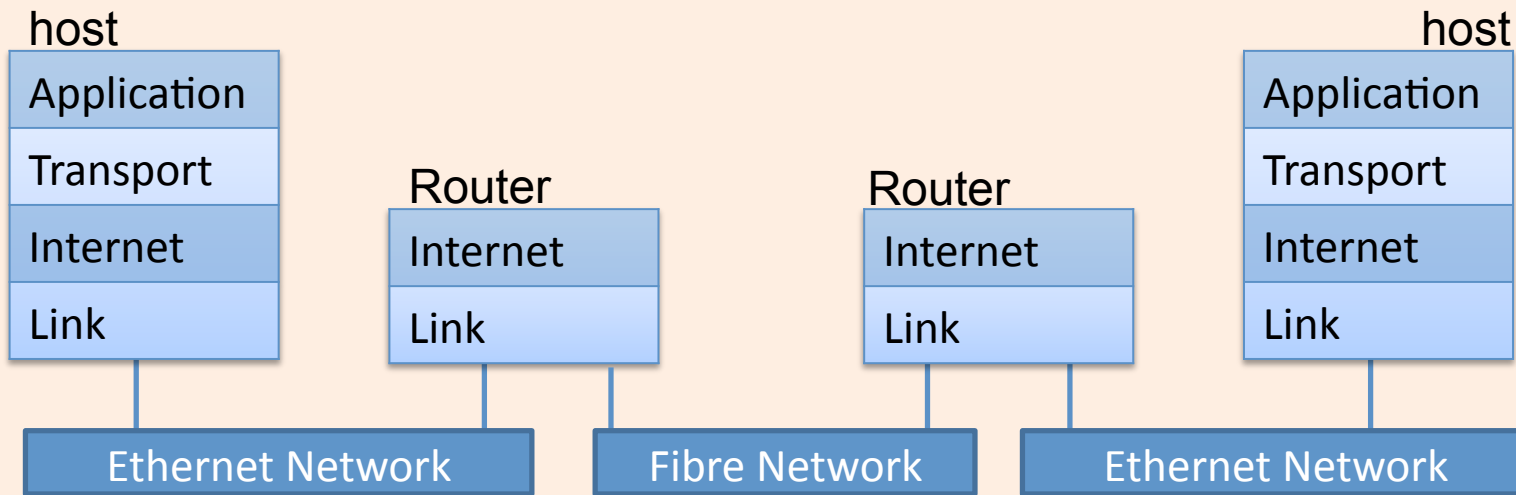
- Check IP header if packet destination is for this host. If yes, strip header and pass to Transport layer
- Otherwise forward packet (routing)

# IPv4 Addresses & Domain Name Service



- IP addresses are 32 bit numbers often written in 4 octets: 128.171.10.13
- Each address is also split into two parts
  - Prefix is the network address
  - Suffix is the host address within that network
- **Domain Name Servers** provide a service that translates more meaningful names to IP addresses
  - Uhunix.hawaii.edu = 128.171.24.197
  - www2.hawaii.edu = 128.171.224.150

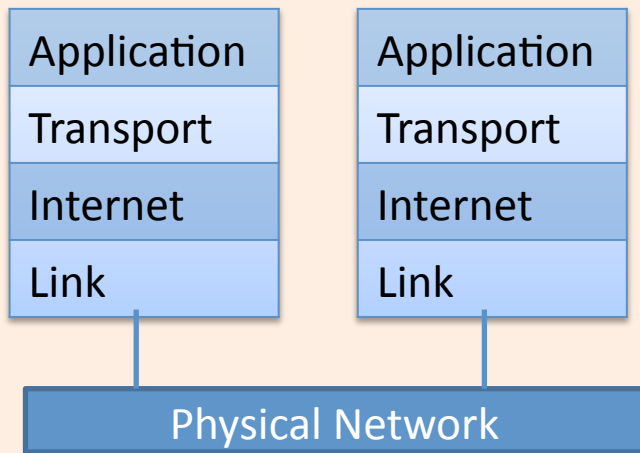
# IPv4 & Inter-network Routing



## For routers

- Examine destination IP address
- Look up routing tables to determine outgoing network
- Pass packet to link layer of that outgoing network
- Best effort delivery – no guarantees!

# Transport Layer

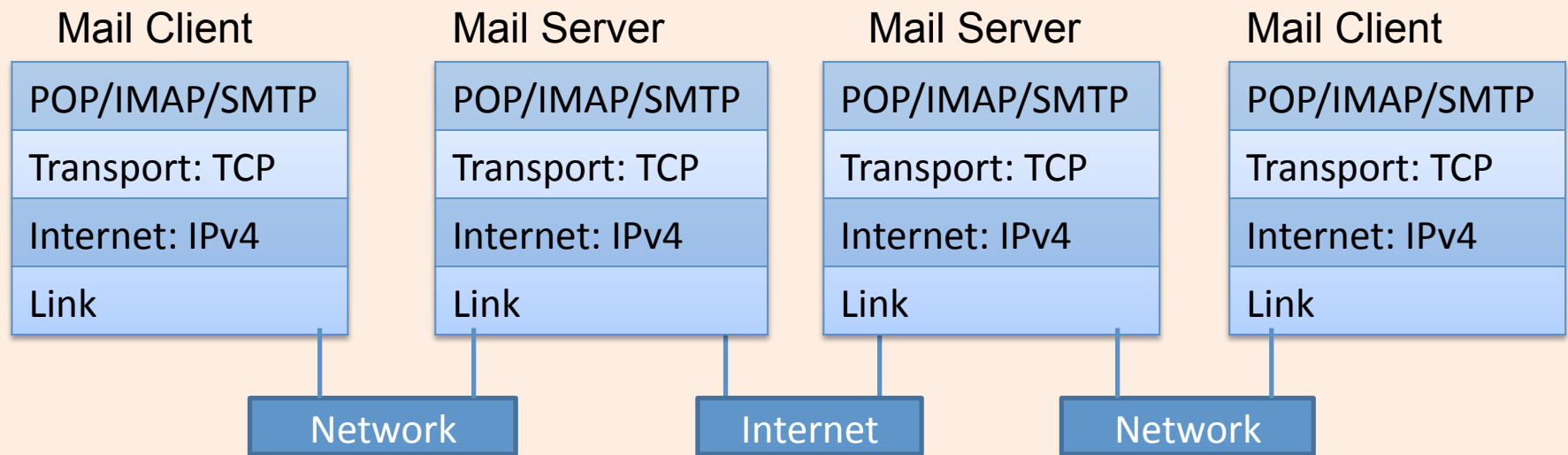


- Eg. TCP (connection-oriented), UDP
- End-to-end message transfer between hosts applications
- Each application on a host is associated with a port number
- IP address + port number will identify an application end-point

TCP provides a reliable communication channel between two host applications by addressing several issues

- Data packets arriving out of order
- Data packets are corrupted
- Same packets arriving more than once
- Some packets are lost/discarded
- Traffic congestion control

# Applications: Email



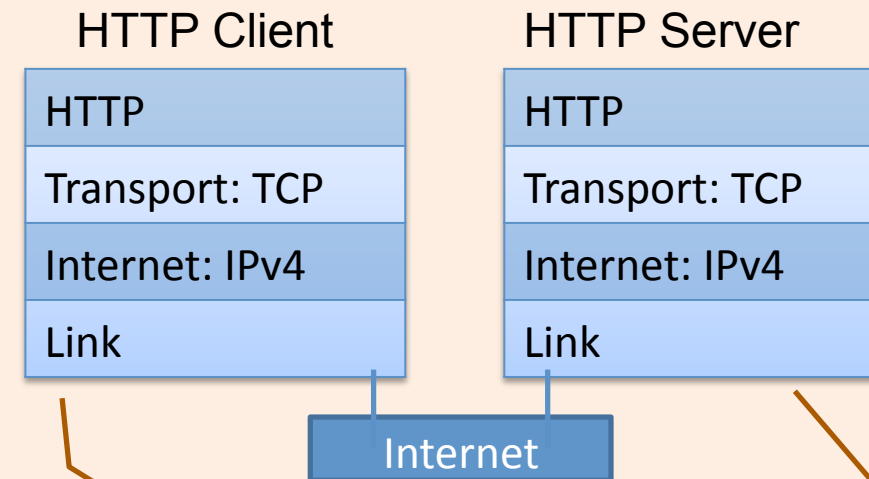
- Your email client program downloads incoming emails from mail server (imap.gmail.com pop.gmail.com)
- Outgoing emails are sent to mail server (smtp.gmail.com)
- Mail servers handle the routing of emails using SMTP protocol which operates on port 25 or 587
  - Lookup IP address of destination hostname in the email address using DNS
  - Relaying email as packets to that IP address

# Sample Email Header

Delivered-To: [strev@guhrelay.hawaii.edu](mailto:strev@guhrelay.hawaii.edu)  
Received: by 10.58.145.6 with SMTP id sq6csp687725veb; Mon, 3 Sep 2012 20:39:01 -0700 (PDT)  
Received: by 10.68.129.38 with SMTP id nt6mr43102232pbb.76.1346729940698; Mon, 03 Sep 2012 20:39:00 -0700 (PDT)  
Return-Path: <[postmaster@laulima.hawaii.edu](mailto:postmaster@laulima.hawaii.edu)>  
Received: from [mta11.its.hawaii.edu](mailto:mta11.its.hawaii.edu) ([mta11.its.hawaii.edu](mailto:mta11.its.hawaii.edu) [128.171.224.147])  
by [mx.google.com](http://mx.google.com) with ESMTPS id px6si25354378pbc.214.2012.09.03.20.38.53  
(version=TLSv1/SSLv3 cipher=RC4-MD5); Mon, 03 Sep 2012 20:39:00 -0700 (PDT)  
Received-SPF: pass ([google.com](http://google.com): domain of [postmaster@laulima.hawaii.edu](mailto:postmaster@laulima.hawaii.edu) designates [128.171.224.58](mailto:128.171.224.58) as permitted sender) client-ip=[128.171.224.58](mailto:128.171.224.58);  
Authentication-Results: [mx.google.com](http://mx.google.com); spf=pass ([google.com](http://google.com): domain of [postmaster@laulima.hawaii.edu](mailto:postmaster@laulima.hawaii.edu) designates [128.171.224.58](mailto:128.171.224.58) as permitted sender)  
smtp.mail=[postmaster@laulima.hawaii.edu](mailto:postmaster@laulima.hawaii.edu)  
MIME-version: 1.0  
Content-type: multipart/mixed;  
boundary="Boundary\_(ID\_3RY8N2VbJHb4tH5siR1eww)"

Received:  
from [pmx11.its.hawaii.edu](mailto:pmx11.its.hawaii.edu) ([pmx11.its.hawaii.edu](mailto:pmx11.its.hawaii.edu) [128.171.224.58]) by  
[mta11.its.hawaii.edu](mailto:mta11.its.hawaii.edu) (Sun Java(tm) System Messaging Server 6.3-11.01 (built Feb 12 2010; 32bit)) with ESMTP id  
<[0M9T0071I3GJ4F40@mta11.its.hawaii.edu](mailto:0M9T0071I3GJ4F40@mta11.its.hawaii.edu)>;  
Mon, 03 Sep 2012 17:38:45 -1000 (HST)  
Received:  
from [kuhi.its.hawaii.edu](mailto:kuhi.its.hawaii.edu) ([kuhi.its.hawaii.edu](mailto:kuhi.its.hawaii.edu) [128.171.25.223])  
by [pmx11.its.hawaii.edu](mailto:pmx11.its.hawaii.edu) (Postfix) with ESMTP id E587118C023; Mon, 03 Sep 2012 17:38:42 -1000 (HST)  
Received:  
from [sak24.its.hawaii.edu](mailto:sak24.its.hawaii.edu) ([sak24.its.hawaii.edu](mailto:sak24.its.hawaii.edu) [128.171.225.199])  
by [kuhi.its.hawaii.edu](mailto:kuhi.its.hawaii.edu) (8.12.10/8.12.6) with ESMTP id q843ccvH023430; Mon, 03 Sep 2012 17:38:38 -1000 (HST)  
Date: Mon, 03 Sep 2012 17:38:33 -1000 (HST)  
From: Dennis Streveler <[strev@hawaii.edu](mailto:strev@hawaii.edu)>  
Cc: "[strev@hawaii.edu](mailto:strev@hawaii.edu)" <[strev@hawaii.edu](mailto:strev@hawaii.edu)>  
Message-id:  
<[112987554.2310.1346729913602.JavaMail.sakai@sak24.its.hawaii.edu](mailto:112987554.2310.1346729913602.JavaMail.sakai@sak24.its.hawaii.edu)>  
Subject: ICS 101 Help: Tuesday lecture -- Everything you THOUGHT you knew  
about NETWORKS and then some  
X-Mailer: sakai-mailsender

# Applications: HTTP



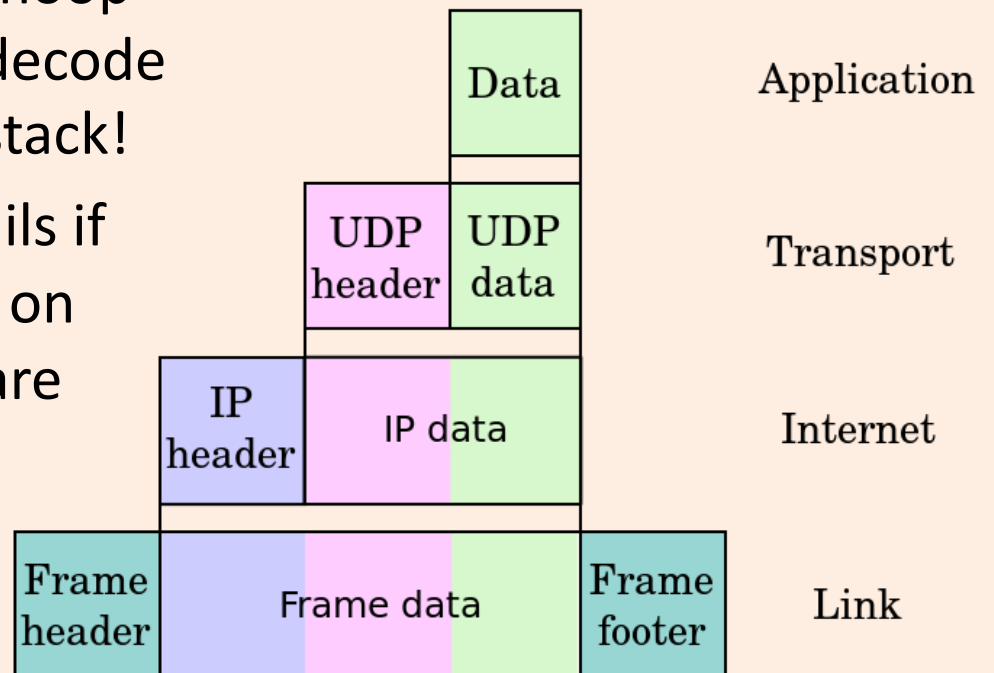
- Hyper-Text Transfer Protocol (port 80)
- Request-response protocol
- When <http://www2.hawaii.edu/~lipyeow/index.html> is entered into a web browser (http client)

```
GET /~lipyeow/index.html HTTP/1.1
host: www2.hawaii.edu
```

```
HTTP/1.1 200 OK
Date: Sun, 02 Sep 2012 00:35:40 GMT
Server: Apache
Last-Modified: Tue, 21 Aug 2012 01:27:18 GMT
ETag: "7d3e8-2950-4c7bc86e86980"
Accept-Ranges: bytes
Content-Length: 10576
Content-Type: text/html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <HTML> ...
```

# Internet Security

- All data transmitted on the network using the protocols described thus far are in plaintext
- Anyone with access to the physical network link can snoop on the bit sequences and decode according to the protocol stack!
- Anyone can read your emails if he/she has access to a link on which your email packets are transmitted
- Use encrypted connections eg. SSL/TLS

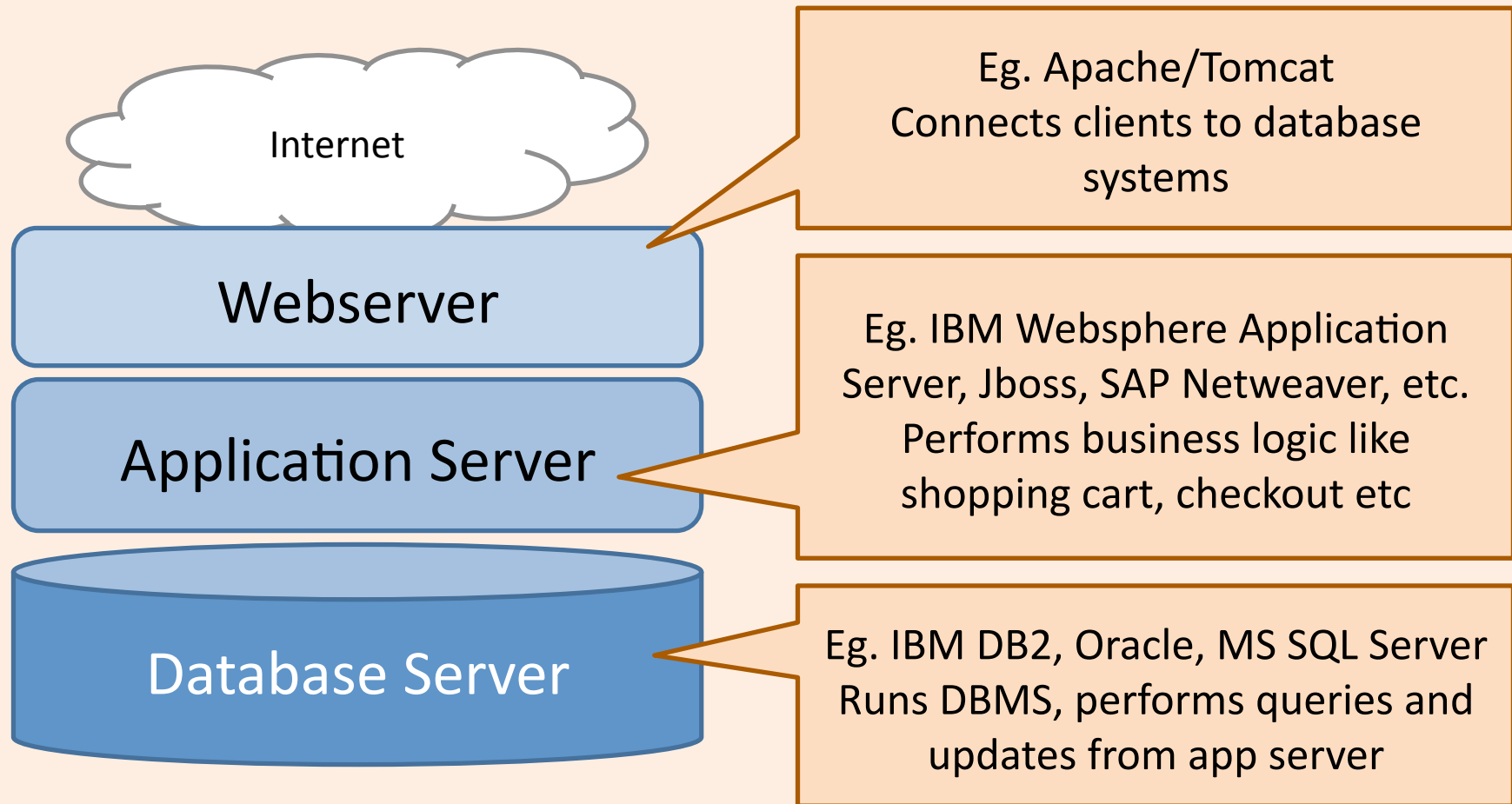




# ICS 321 Data Storage & Retrieval

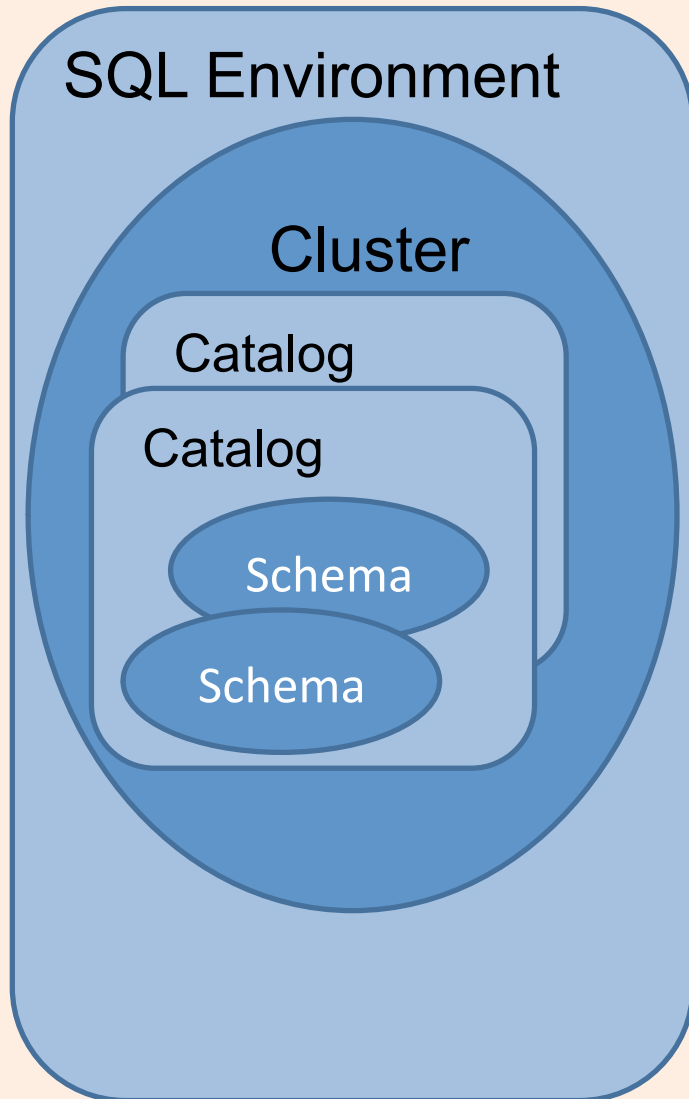
## SQL in a Server Environment

# Three Tier Architecture



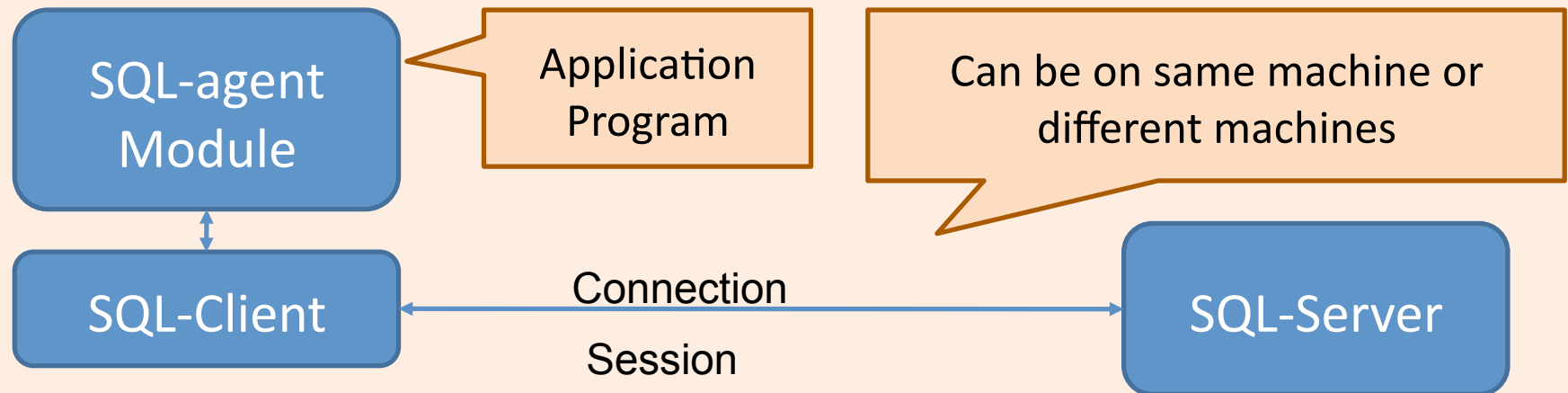
- Commonly used in large internet enterprises

# SQL Environment



- Schemas : tables, views, assertions, triggers
  - **CREATE SCHEMA** <schema name>
  - Your login id is your default schema
  - **SET SCHEMA** <schema>
  - A fully qualified table name is <schema>.<table>
- Catalogs : collection of schemas
  - Corresponds to “databases” in DB2
- Clusters : collection of catalogs
  - Corresponds to “database instance” in DB2

# Client-Server Model



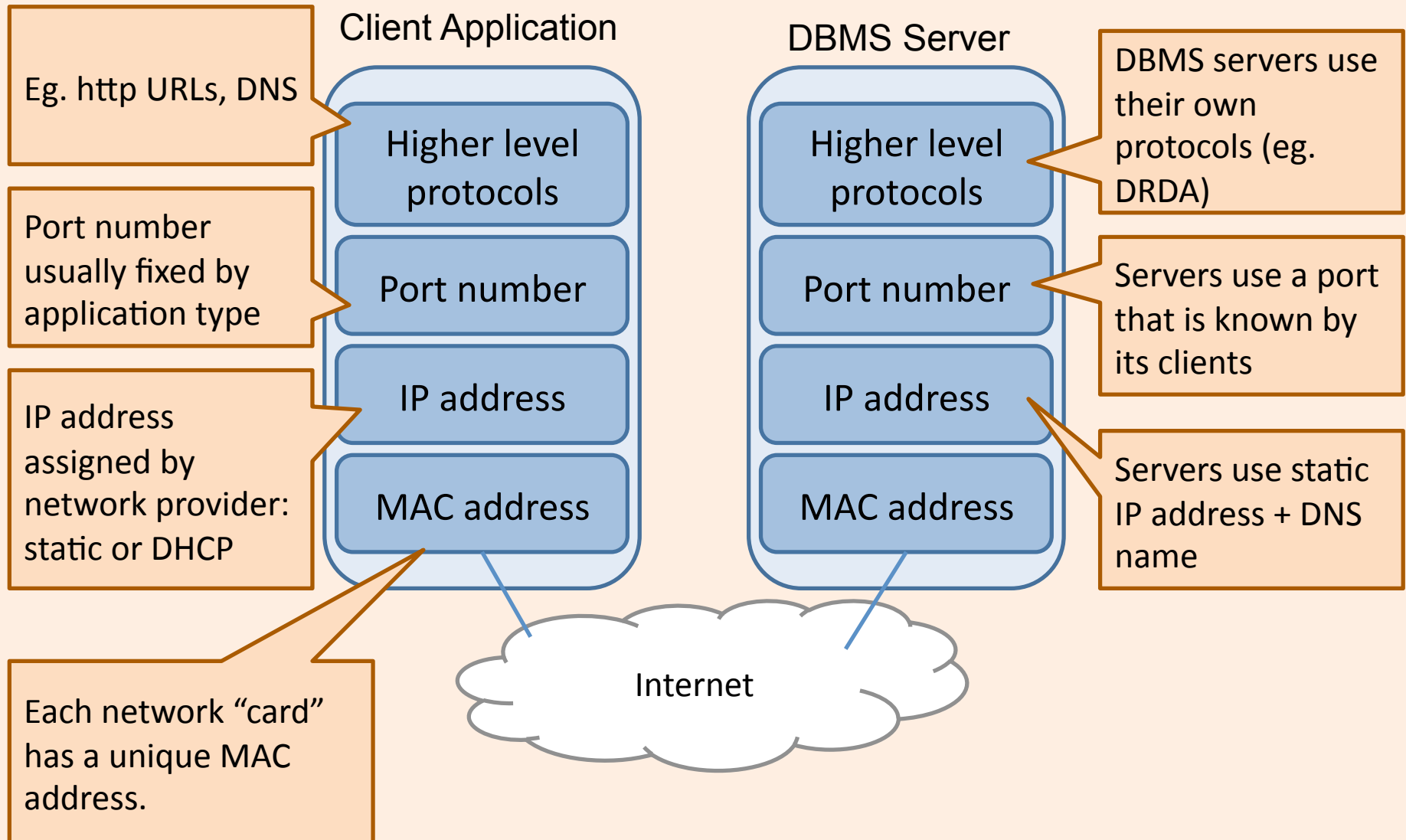
- **CONNECT TO <server> AS <connection name> AUTHORIZATION**
- **DISCONNECT/CONNECT RESET/TERMINATE**
- Session – SQL operations performed while a connection is active
- Programming API
  - Generic SQL Interface
  - Embedded SQL in a host language
  - True Modules. Eg. Stored procedures.

# SQL & Other Programming Languages

Two extremes of the integration spectrum:

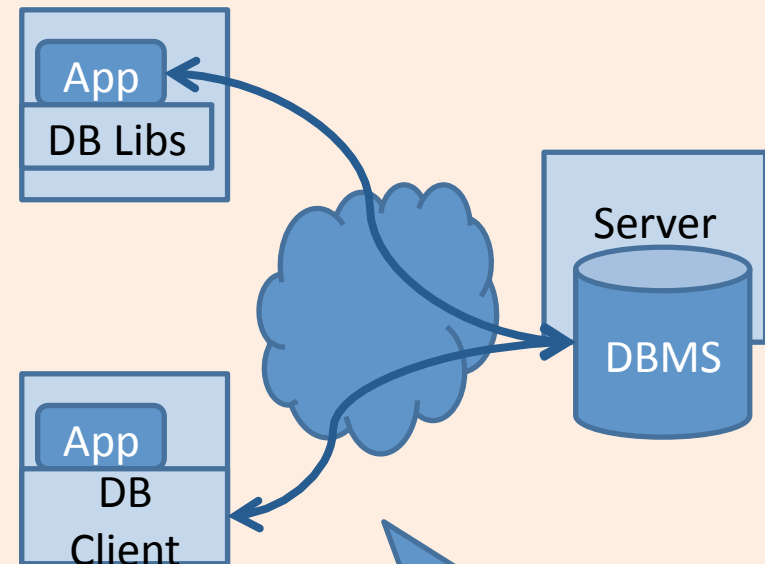
- Highly integrated eg. Microsoft linq
  - Compiler checking of database operations
- Loosely integrated eg. ODBC & JDBC
  - Provides a way to call SQL from host language
  - Host language compiler doesn't understand database operations.
- Requirements:
  - Perform DB operations from host language
  - DB operations need to access variables in host language

# Networking Basics



# Remote Client Access

- Applications run on a machine that is separate from the DB server
- DBMS “thin” client
  - Libraries to link your app to
  - App needs to know how to talk to DBMS server via network
- DBMS “full” client layer
  - Need to pre-configure the thick client layer to talk to DBMS server
  - Your app talks to a DBMS client layer as if it is talking to the server



What information is needed for 2 machines to talk over a network ?

# Configuring DBMS Client Layer

- Tell the client where to find the server

```
db2 CATALOG TCPIP NODE mydbsrv  
REMOTE 123.3.4.12 SERVER 50001
```

Give a name for  
this node

Specify the IP  
address/  
hostname and the  
port number of  
the DB server  
machine

- Tell the client where to find the server

```
db2 CATALOG DATABASE bookdb AS  
mybookdb AT NODE mydbsrv
```

Give a local alias  
for the database

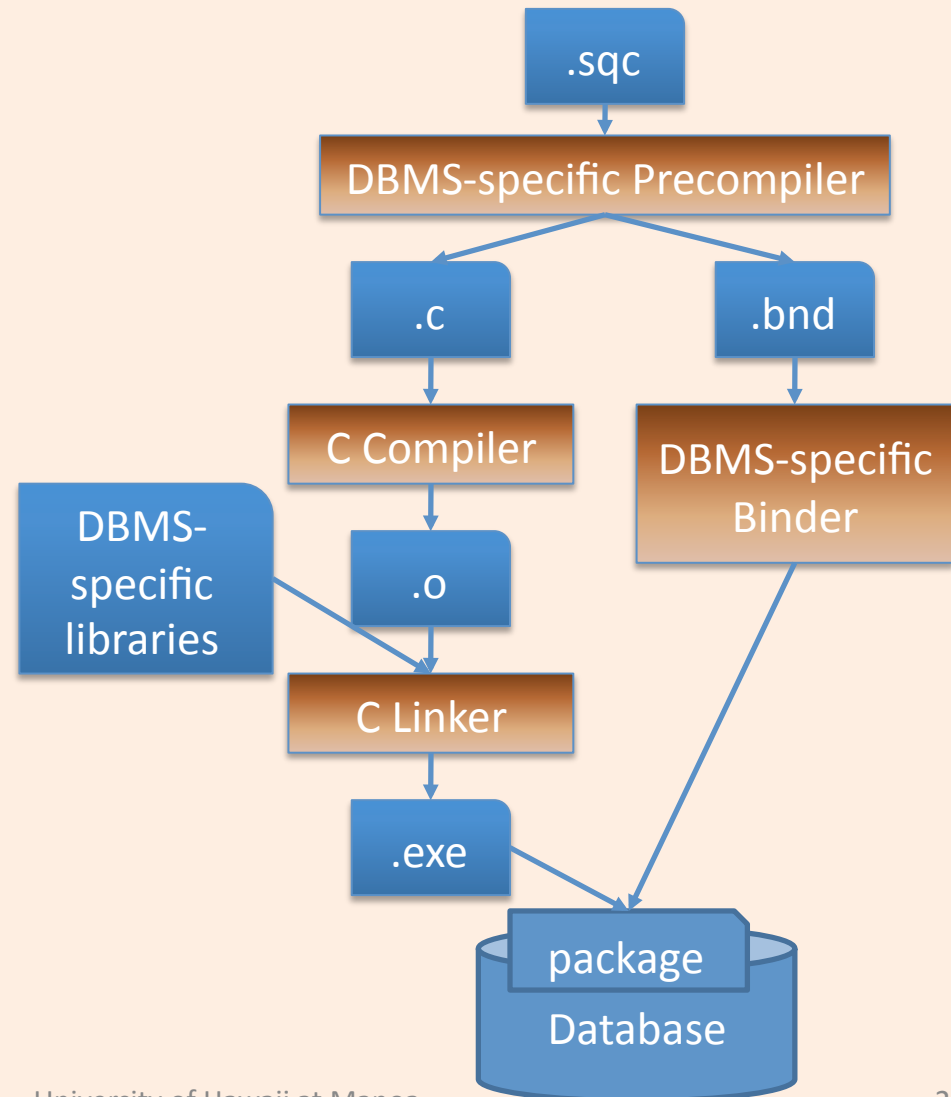
Specify the name of the  
node that is associated  
with this database

Specify the name  
of the database  
on the server



# Embedded SQL in C Programs

- DBMS-specific Preprocessor translates special macros to DB-specific function calls
- Pre-processor needs access to DBMS instance for validation.
- Executable needs to be bound to a specific database in a DBMS in order to execute



# Connecting SQL & Host Language

- Need a way for host language to **get data** from SQL environment
- Need a way to **pass values** from host language to SQL environment
- Shared variables
  - **DECLARE SECTION**
  - In SQL, refer using :Salary, :EmployeeNo

```
EXEC SQL BEGIN DECLARE SECTION;  
char EmployeeNo[7];  
char LastName[16];  
double Salary;  
short SalaryNI;  
EXEC SQL END DECLARE SECTION;
```

# An Example of Embedded SQL C Program

```
#include <stdio.h>
#include <string.h>
#include <sql.h>
int main()
{
// Include The SQLCA Data Structure Variable
EXEC SQL INCLUDE SQLCA;

// Define The SQL Host Variables Needed
EXEC SQL BEGIN DECLARE SECTION;
char EmployeeNo[7];
char LastName[16];
double Salary;
short SalaryNI;
EXEC SQL END DECLARE SECTION;

// Connect To The Appropriate Database
EXEC SQL CONNECT TO SAMPLE USER
db2admin USING ibmdb2;

// Declare A Static Cursor
EXEC SQL DECLARE C1 CURSOR FOR
SELECT EMPNO, LASTNAME, DOUBLE(SALARY)
FROM EMPLOYEE
WHERE JOB = 'DESIGNER';

// Open The Cursor
EXEC SQL OPEN C1;
```

# An Example of Embedded SQL C Program

```
// If The Cursor Was Opened Successfully,
while (sqlca.sqlcode == SQL_RC_OK)
{
    EXEC SQL FETCH C1 INTO :EmployeeNo,
                          :LastName, :Salary, :SalaryNI;

    // Display The Record Retrieved
    if (sqlca.sqlcode == SQL_RC_OK)
    {
        printf("%-8s %-16s ", EmployeeNo,
                LastName);
        if (SalaryNI >= 0)
            printf("%lf\n", Salary);
        else
            printf("Unknown\n");
    }
}
```

```
// Close The Open Cursor
EXEC SQL CLOSE C1;
// Commit The Transaction
EXEC SQL COMMIT;
// Terminate The Database Connection
EXEC SQL DISCONNECT CURRENT;
// Return Control To The Operating System
return(0);
}
```

- A cursor is an iterator for looping through a relation instance.
- Why is a cursor construct necessary ?

# Updates

- SQL syntax except **where** clause require **current of <cursor>**

```
EXEC SQL BEGIN DECLARE  
SECTION;  
int certNo , worth ;  
char execName[31],  
    execName[31],  
    execAddr [256],  
    SQLSTATE [6];  
EXEC SQL END DECLARE  
SECTION;
```

```
EXEC SQL DECLARE execCursor CURSOR FOR  
    MovieExec;  
EXEC SQL OPEN execCursor  
while (1) {  
    EXEC SQL FETCH FROM execCursor  
        INTO :execName, :execAddr, :certNo, :worth;  
    if (NO_MORE_TUPLES) break;  
    if ( worth < 1000)  
        EXEC SQL DELETE FROM MovieExec  
            WHERE CURRENT OF execCursor;  
    else  
        EXEC SQL UPDATE MovieExec  
            SET netWorth=2*netWorth  
            WHERE CURRENT OF execCursor;  
}  
EXEC SQL CLOSE execCursor
```

# Static vs Dynamic SQL

- Static SQL refers to SQL queries that are completely specified at compile time. Eg.
- Dynamic SQL refers to SQL queries that are not completely specified at compile time. Eg.

```
// Declare A Static Cursor  
EXEC SQL DECLARE C1 CURSOR FOR  
SELECT EMPNO, LASTNAME,  
       DOUBLE(SALARY)  
FROM EMPLOYEE  
WHERE JOB = 'DESIGNER';
```

```
strcpy(SQLStmt, "SELECT * FROM  
EMPLOYEE WHERE JOB=");  
strcat(SQLStmt, argv[1]);  
EXEC SQL PREPARE SQL_STMT  
FROM :SQLStmt;  
EXEC SQL EXECUTE SQL_STMT;
```