i206: Lecture 21: Review for Test 3

Tapan Parikh

Spring 2013

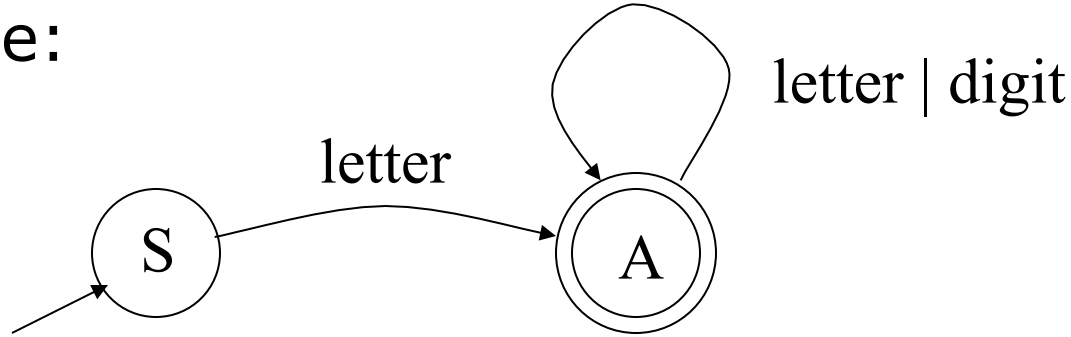
Some slides courtesy Marti Hearst, John Chuang and others

Topics

- Regular expressions / Finite Automata
- B-Trees
- Operating Systems
- Networking
- Databases
- Distributed Systems
  - (hadoop, map reduce)
- Cryptography

Language of a FA

- Language of finite automaton M: set of all strings accepted by M
- Example:



- Which of the following are in the language?
  - x, tmp2, 123, a?, 2apples

- A language is called a regular language if it is recognized by some finite automaton

Regex for Dollars

- No commas

  \$[0-9]+(\.[0-9][0-9])?

- With commas

  \$[0-9][0-9]?[0-9]?(,[0-9][0-9][0-9])*(\.[0-9][0-9])?

- With or without commas

  \$[0-9][0-9]?[0-9]?((,[0-9][0-9][0-9])*| [0-9]*)
  (\.[0-9][0-9])?

Trees

- Simplest: binary tree
- More common for disk-based storage: B-trees
- Pros:
  - Solves the prefix problem (terms starting with *hyp*)
- Cons:
  - Slower: O(log *N*) lookup [for *balanced* tree]
  - Rebalancing binary trees is expensive
    - But B-trees mitigate the rebalancing problem

B^+-Trees

One most commonly implemented form of the B-Tree is the B^+-Tree.

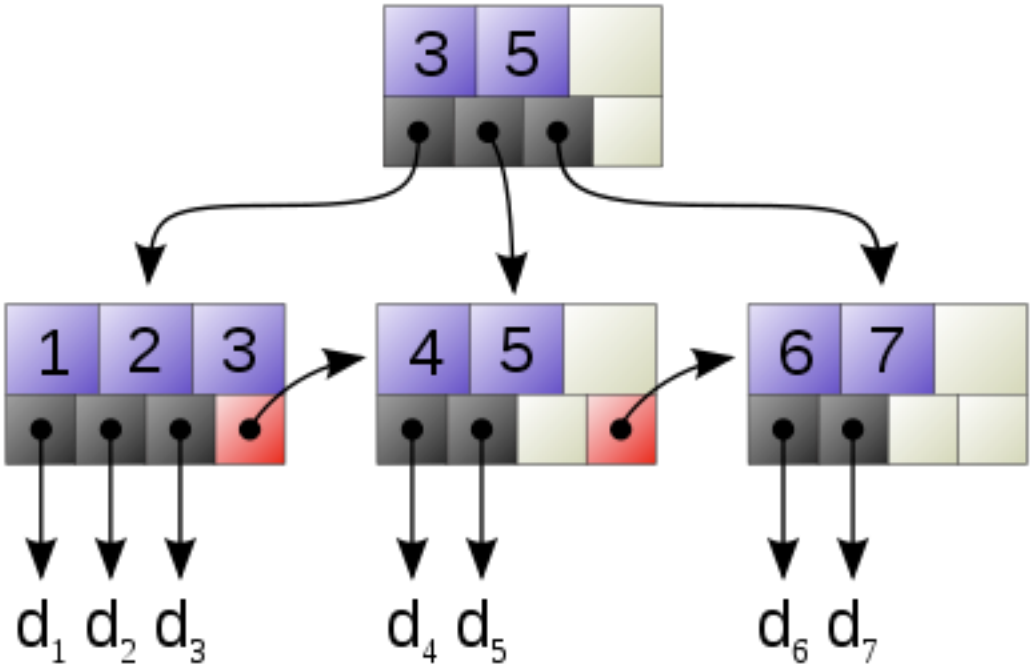
Internal nodes of the B^+-Tree do not store records -- only key values to guide the search.

Leaf nodes store records or pointers to records.

A leaf node may store more or fewer records than an internal node stores keys.

Leaf nodes connected in a linked list for iteration

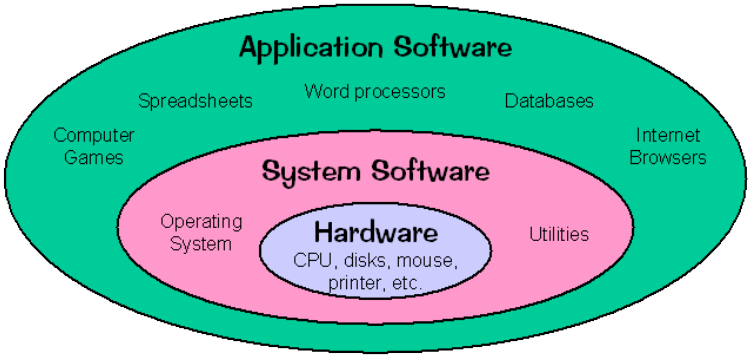
B+tree: Can have more then 2 children



A simple B+ tree example linking the keys 1–7 to data values d1-d7. Note the linked list (red) allowing rapid in-order traversal.

http://www.thefullwiki.org/B%2B_tree

Operating Systems



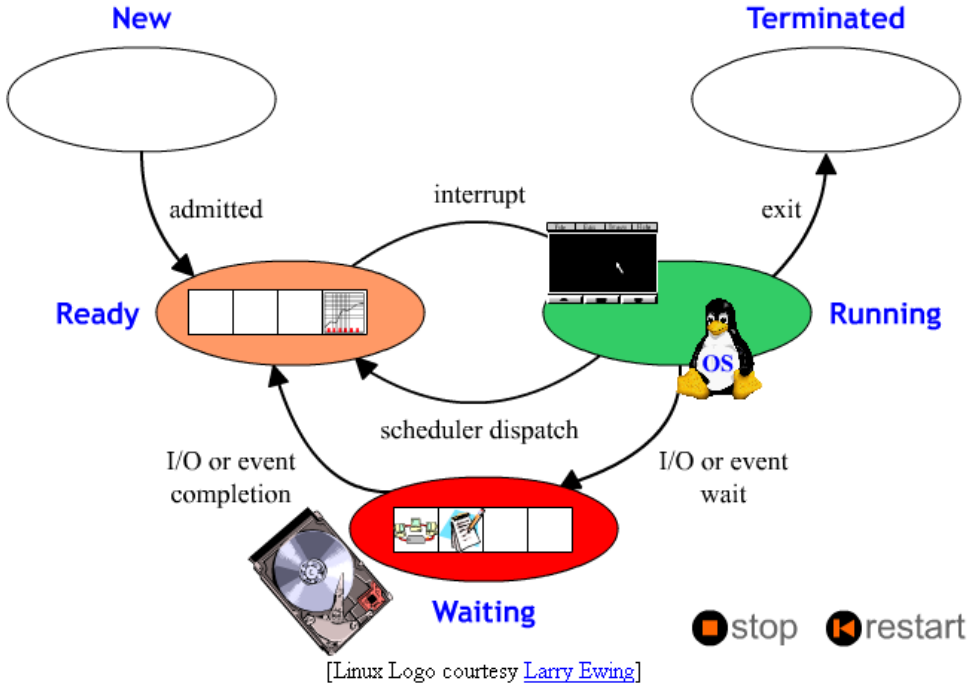
- Main Roles:
  - Allow user applications to run on a given machine's hardware
  - Act as a "traffic cop" for allocating resources.
- Resources
  - CPU, I/O devices, Memory, Files, Allocation
- Processes
  - versus Programs
  - Scheduling
  - Synchronization
- Memory Management
- File Management

Image from http://courses.cs.vt.edu/~csonline/OS/Lessons/Introduction/index.html

Programs vs. Processes

- Program
  - The code, both in human and machine-readable form
- Process
  - A running program is a process.
  - It is allocated a number of resources:
    - Registers in the CPU
    - Some part of the CPU's RAM
    - I/O devices (the monitor, files in the file system)
- Several copies of the same program can be running as different processes.
  - But data values, current Program Counter location, etc., will differ among the different processes.

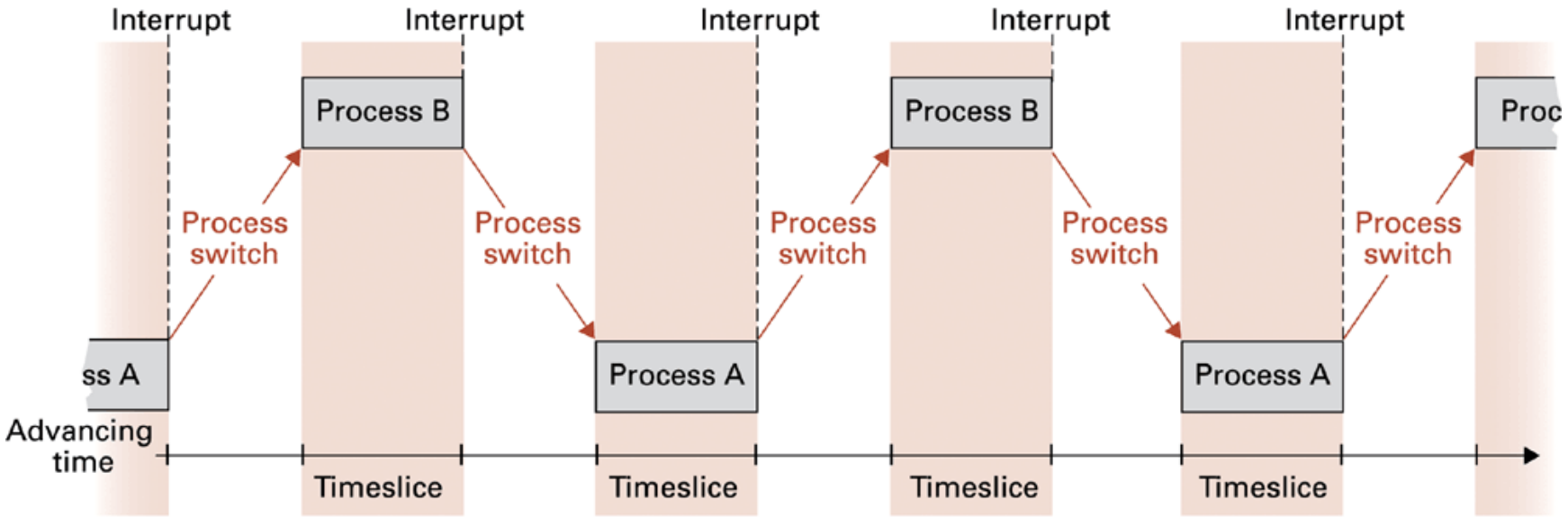
Process State



- As a process executes, it changes *state*
  - **new**: The process is being created.
  - **running**: Instructions are being executed.
  - **waiting**: The process is waiting for some event to occur.
  - **ready**: The process is waiting to be assigned to a process.
  - **terminated**: The process has finished execution.

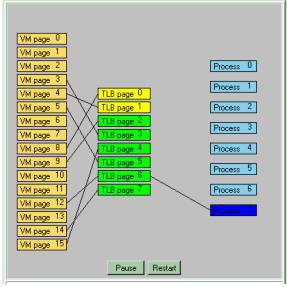
Resource Sharing

- Many processes share hardware resources
  - CPU, I/O (mouse, keyboard, monitor, disk, …)
- Processes must take turns
  - Context switch (aka process switch)
- Operating system serves as coordinator/ scheduler



11

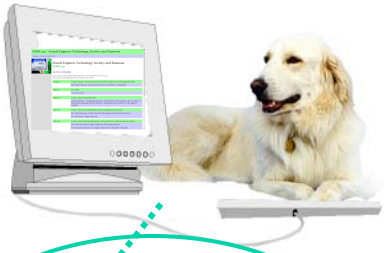
Virtual Memory



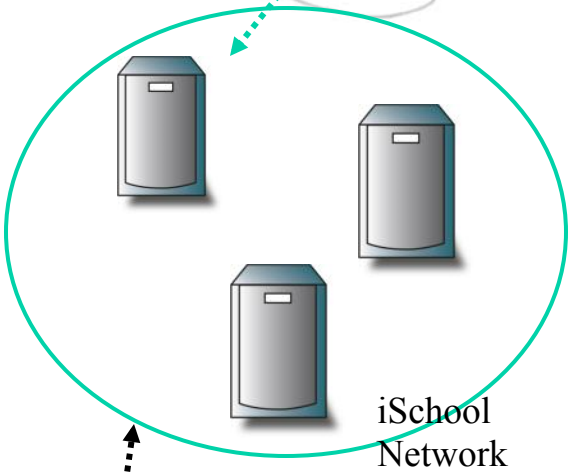
- Allows a process to run even if the full amount of needed memory is not currently available.
  - **Swapping**
    - Moving a page from memory to disk, and vice versa
    - Can also refer to moving an entire process into or out of RAM memory
    - A page fault has to be handled by the Scheduler – so slow (accessing the disk resource; can cause a context switch)
  - **Thrashing**
    - When there isn't enough room in main memory for all the pages that the process needs at the same time
    - The process spends all its time page faulting, swapping processes into and out of memory, and gets little work done.

VM simulation: http://courses.cs.vt.edu/~csonline/OS/Lessons/VirtualMemory/index.html

How Does the WWW Work?

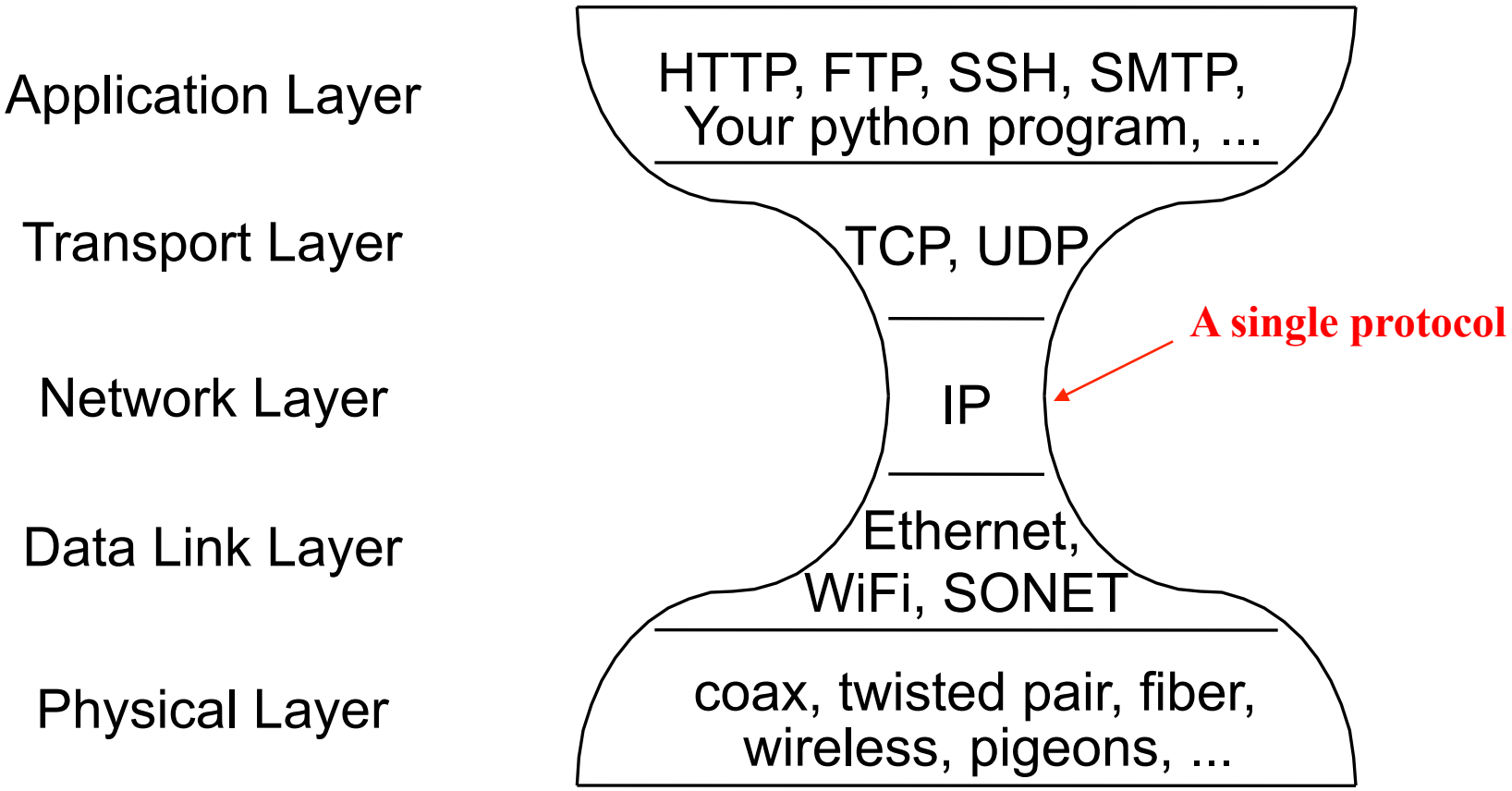


- How does the computer at Oski's desk figure out where the i141 web pages are?
- In order for him to use the WWW, Oski's computer must be connected to another machine acting as a web server (via his ISP).
- This machine is in turn connected to other computers, some of which are **routers**.

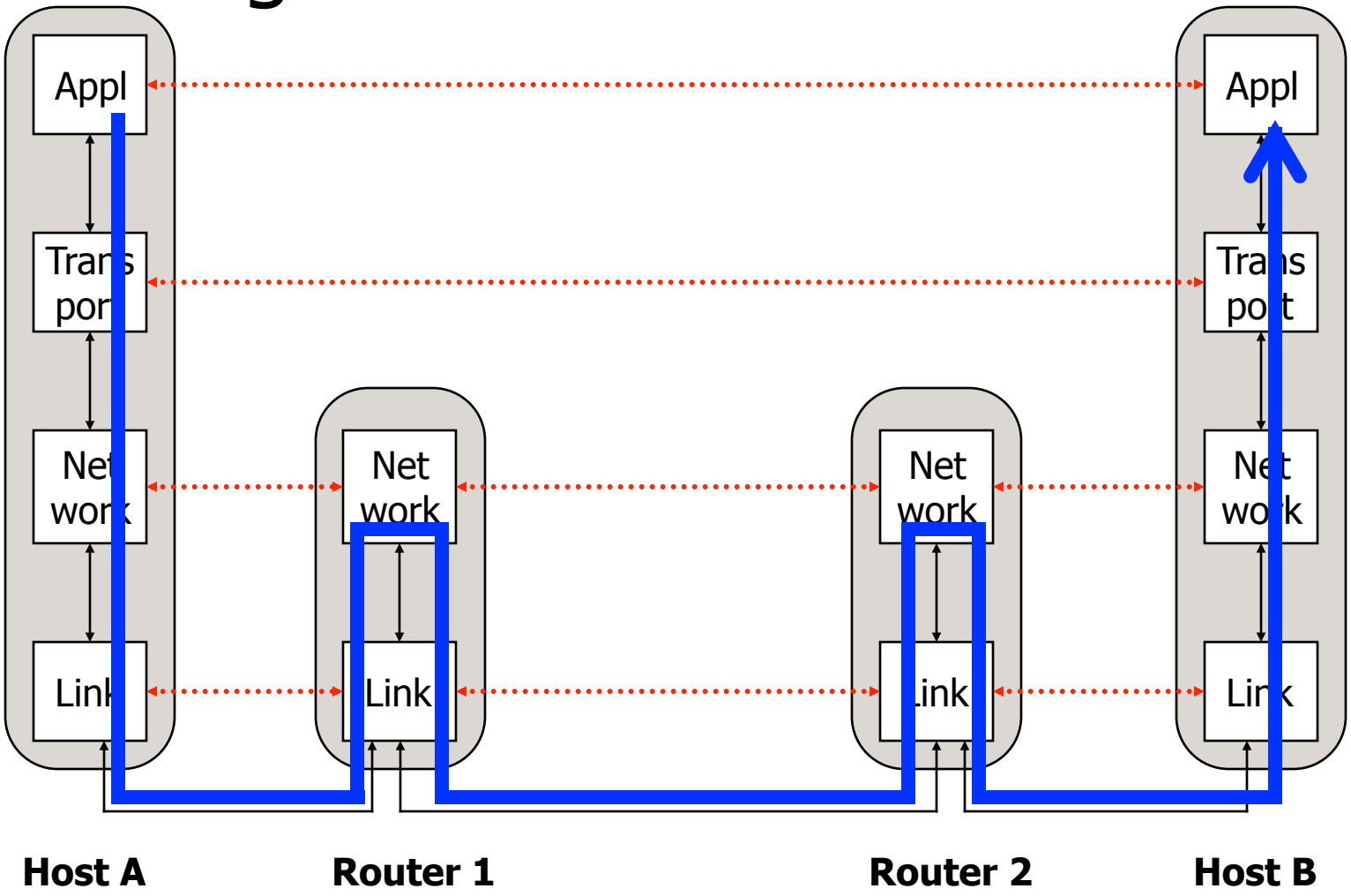


- Routers figure out how to move information from one part of the network to another.
- There are many different possible routes.

The “IP Hourglass”

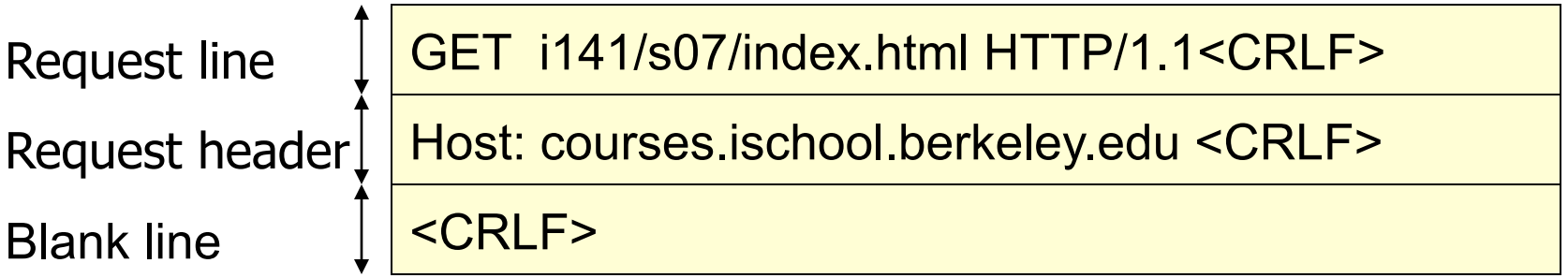


Message Flow



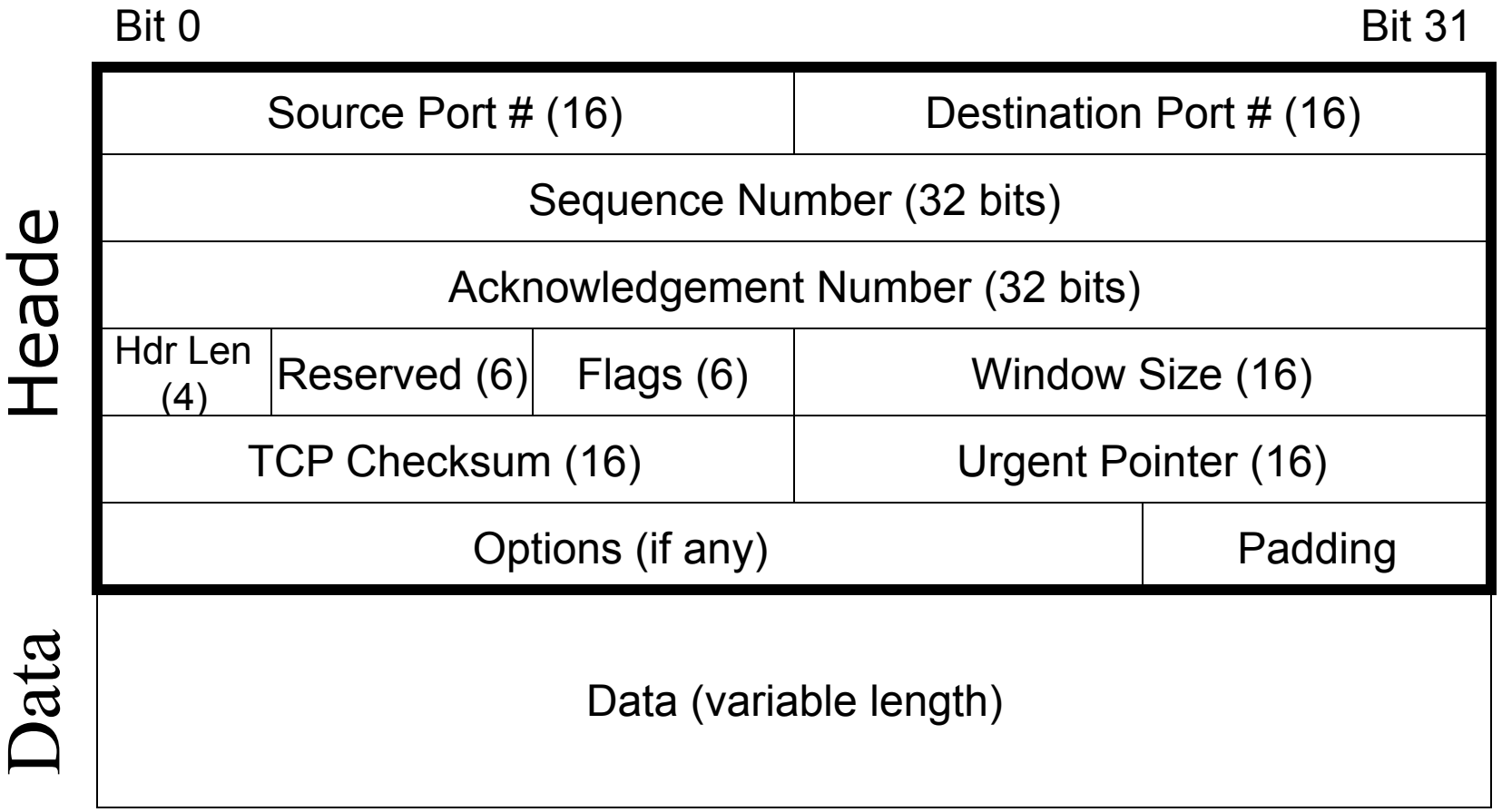
HTTP Request: Example

This information is received by the web server at www.ischool.berkeley.edu :

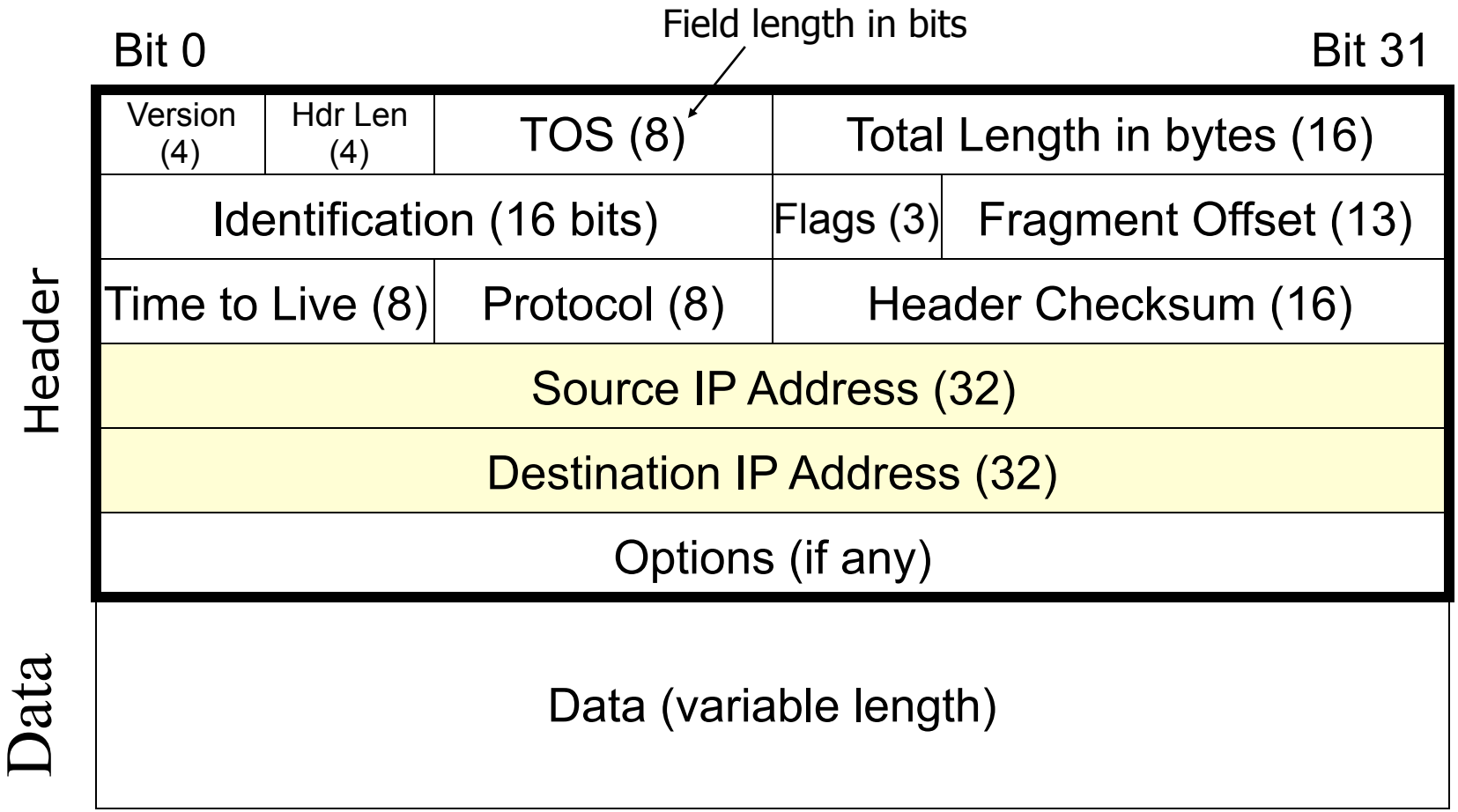


Because HTTP is built on TCP/IP, the web server knows which IP address to send the contents of the web page back to.

TCP Segment Format



IP Packet Format (v4)



What is a database?

- Broadly speaking – any collection of data might be called a database
- Normally, though, certain characteristics of the data and how it is stored distinguish a database from any collection of data

What's a DBMS

- It maintains **Metadata** about the database
  - Data about data
    - In DBMS this means all of the characteristics describing the attributes of an entity, E.G.:
      - name of attributes
      - data type of attributes
      - size of the attributes
      - format or special characteristics
  - Characteristics of tables or 'relations'
    - name, content, notes, etc.

Why use a DBMS?

- You don’t need to write all the code to manage your data
- It will gracefully scale to VERY large collections of data
- It will support transactions that are
  - Atomic (all or nothing)
  - Consistent (from valid state to valid state)
  - Isolated (no interference from concurrent use)
  - Durable (once committed is part of DB)
- Easy to port data to other DBMS or files

SQL

- **S**tructured **Q**uery **L**anguage
- Used for both Database Definition, Modification and Querying
- Basic language is standardized across relational DBMS's. *Each system may have proprietary extensions to standard*
- A few commands are are used, but there are myriad options

Create Table

- CREATE TABLE table-name (attr1 attr-type PRIMARYKEY, attr2 attr-type,…,attrN attr-type);
  - Adds a new table with the specified attributes (and types) to the database.

- In MySQL (5.5+) and SQLite3
  - CREATE TABLE newtablename AS SELECT …
    - Creates new table with contents from SELECT command including data types

INSERT

- INSERT INTO table-name (col1, col2, col3, …, colN) VALUES (val1, val2, val3,…, valN);
- INSERT INTO table-name (col1, col2, col3, …, colN) SELECT…
- Column list is optional, if omitted assumes all columns in table definition and order

SELECT

- Syntax:

```
SELECT  [DISTINCT] attr1, attr2,…, attr3 FROM rel1 r1,
   rel2 r2,… rel3 r3 WHERE condition1 {AND | OR}
   condition2  ORDER BY attr1 [DESC], attr3 [DESC]
```

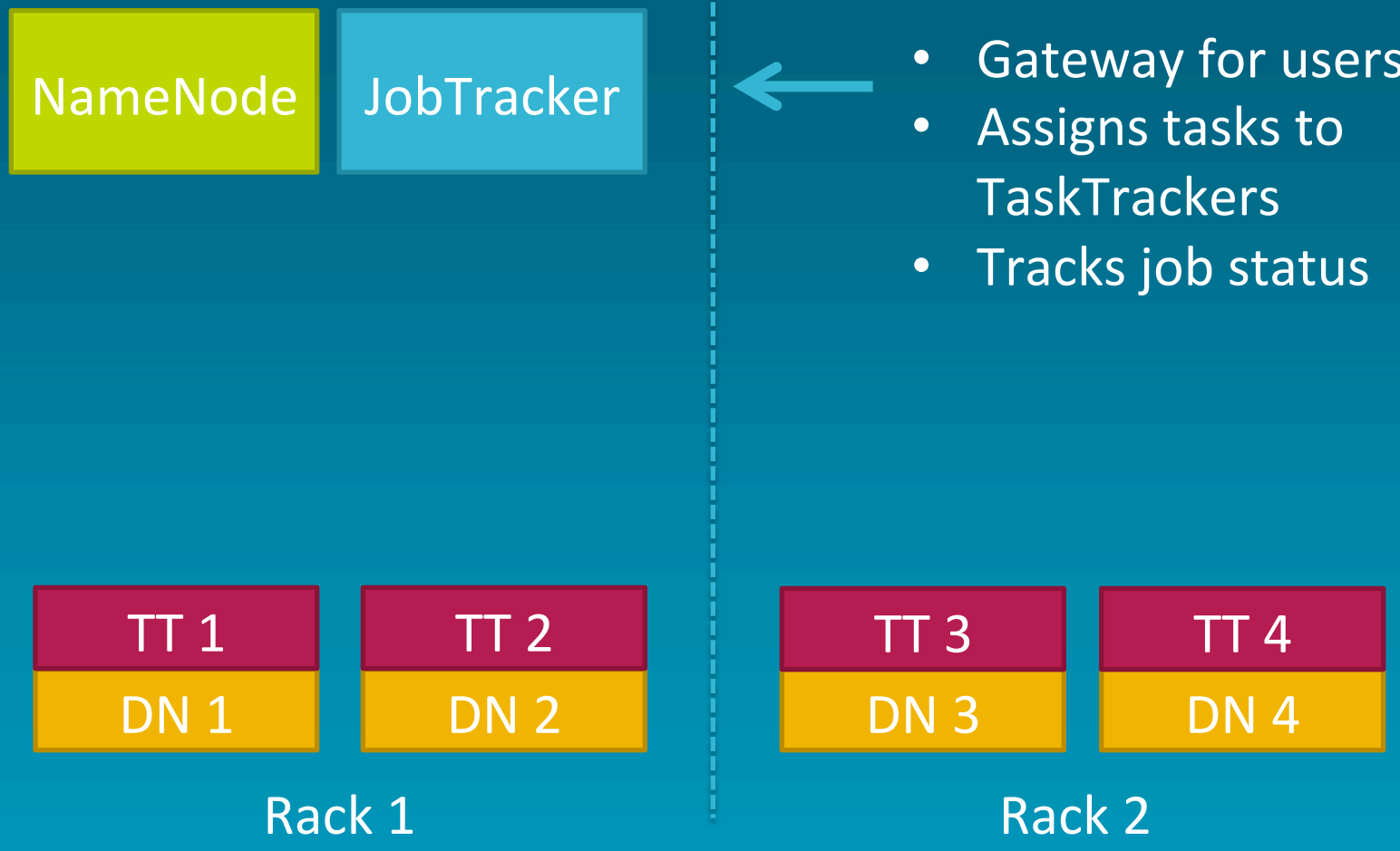
SELECT Conditions

- = equal to a particular value
- >= greater than or equal to a particular value
- > greater than a particular value
- <= less than or equal to a particular value
- <> not equal to a particular value
- LIKE “%term%” (may be other wild cards in other systems)
- IN (“opt1”, “opt2”,…,”optn”)
- BETWEEN val1 AND val2
- IS NULL

Using Aggregate functions

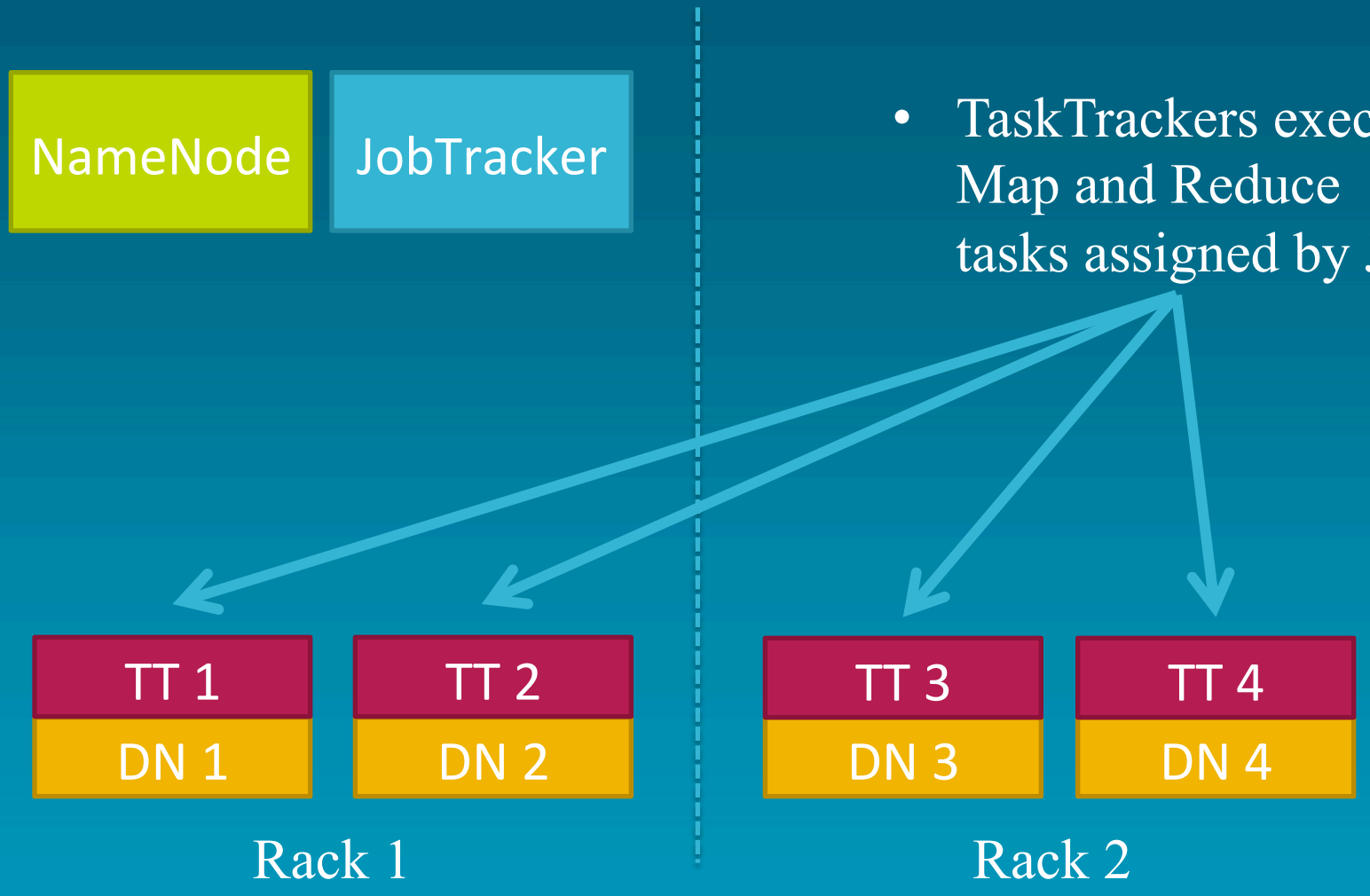
- SELECT attr1, Sum(attr2) AS name
  FROM tab1, tab2 ...
  GROUP BY attr1, attr3 HAVING condition;

MapReduce Architcture

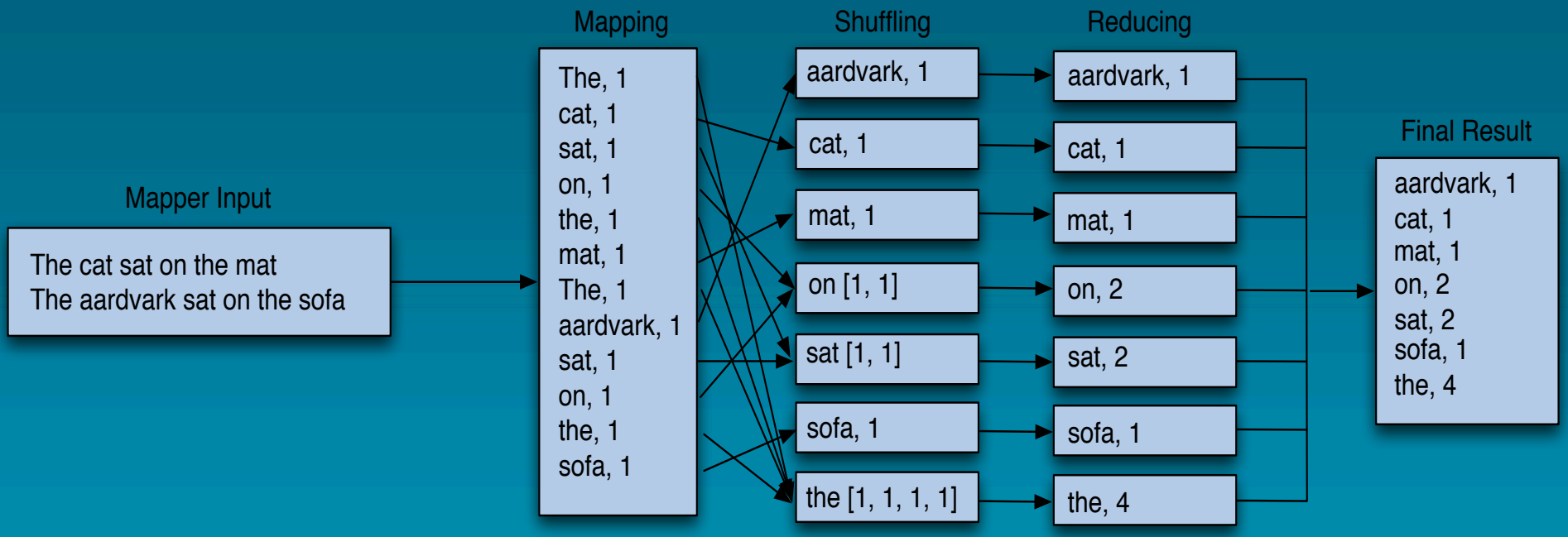


- Gateway for users
- Assigns tasks to TaskTrackers
- Tracks job status

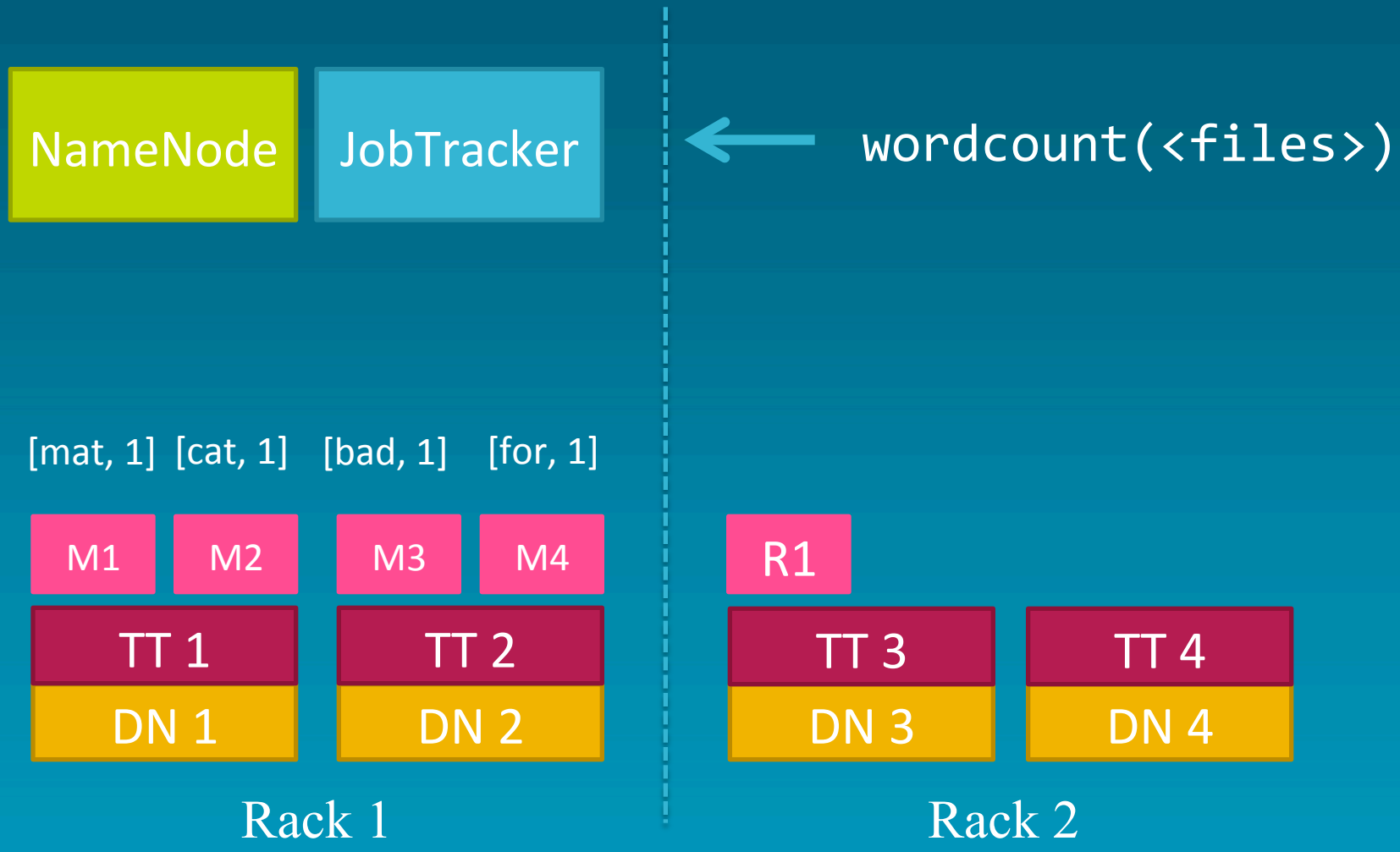
MapReduce Architcture



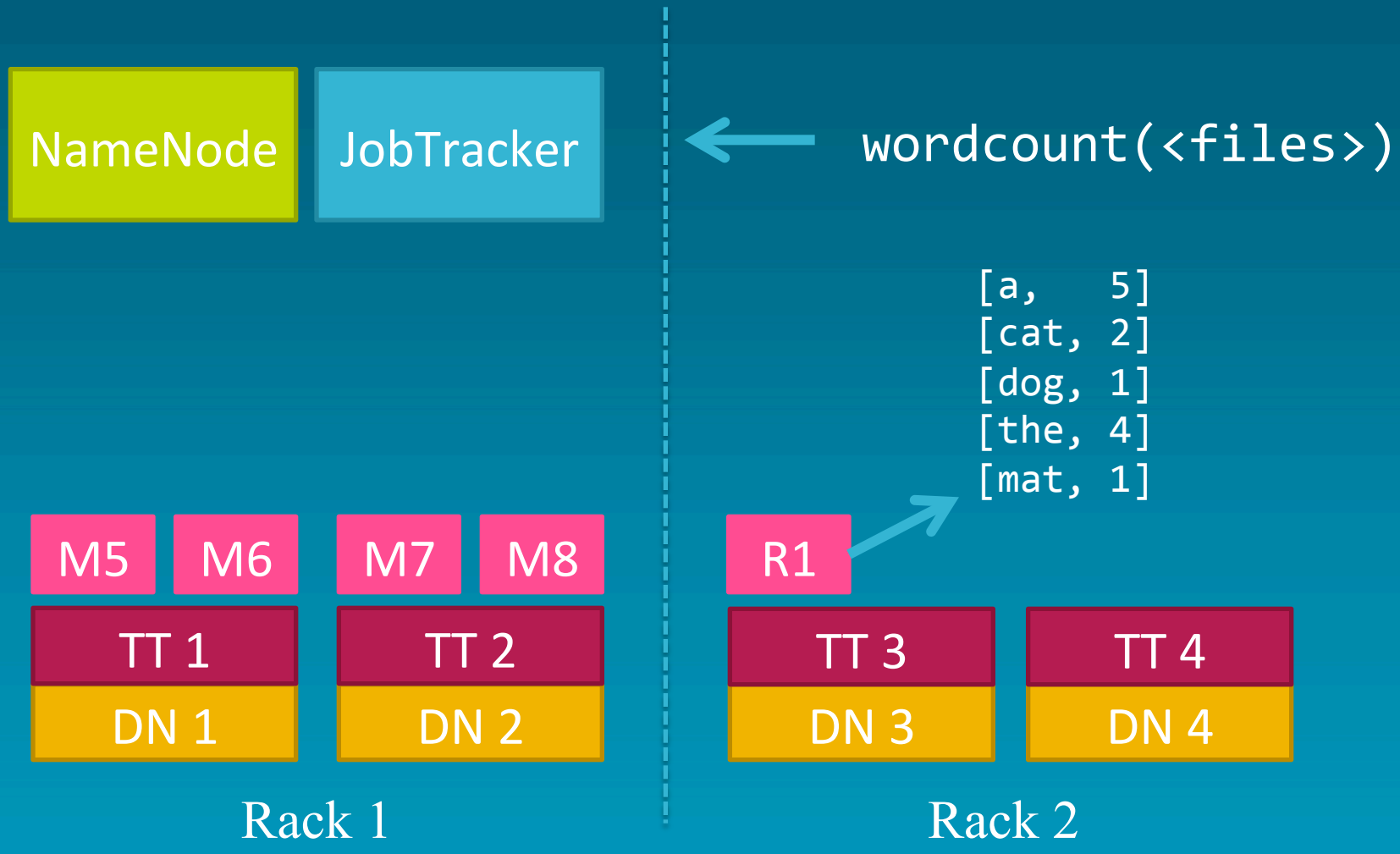
Word Count Example



MapReduce Architcture



MapReduce Architcture

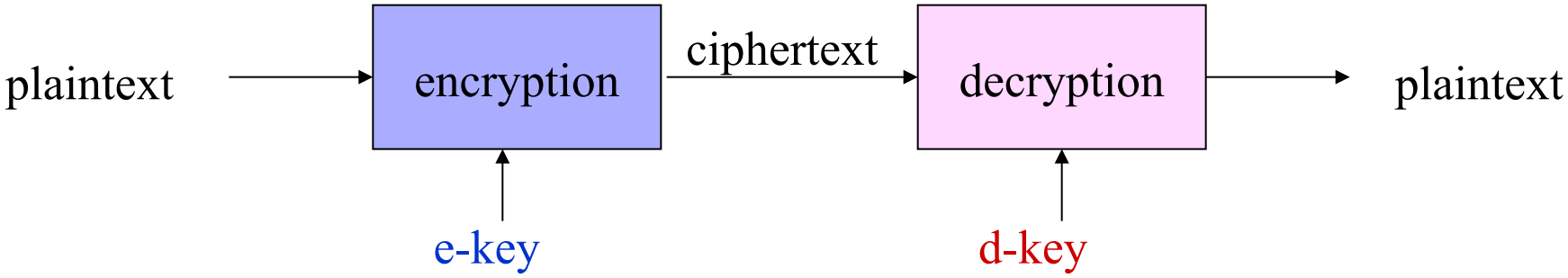


Example of Hadoop Programming

- Intuition: design <key, value>
- Assume each node will process a paragraph…
- Map:
  - What is the key?
  - What is the value?
- Reduce:
  - What to collect?
  - What to reduce?

Slide adapted from Xiaoxiao Shi, Guan Wang

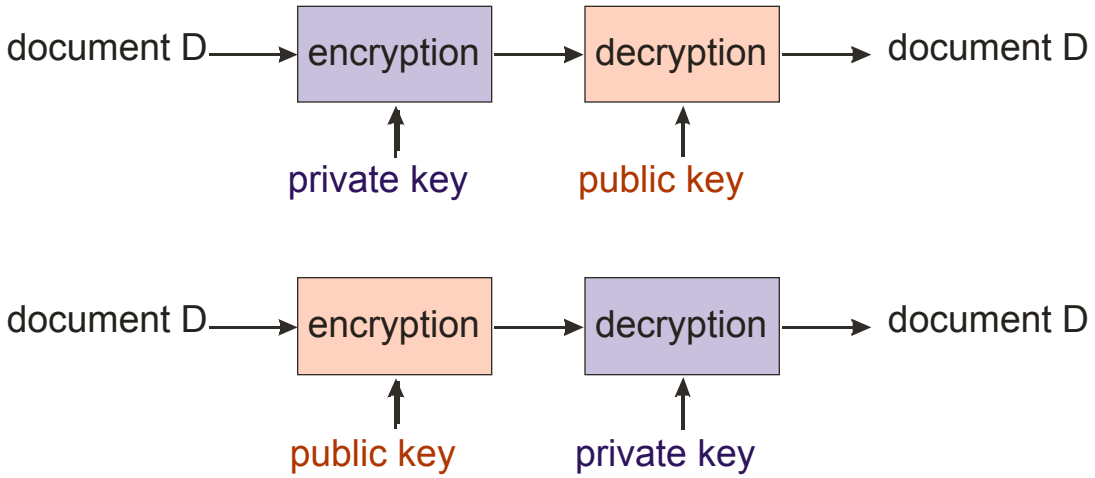
Encryption



- Encryption/decryption algorithms are published
- Encryption/decryption keys are kept secret
- Symmetric cryptography
  - e-key = d-key
  - Principals need to share the symmetric key, and keep it secret
- Asymmetric (public-key) cryptography
  - e-key != d-key
  - One key made public; the other kept private

Asymmetric Cryptography

- Each principal has public key K and private key K^{-1}
- K^{-1} is kept secret, and cannot be deduced from K
- K is made available to all
- Encryption and decryption with K and K^{-1} are commutative: $\{\{D\}K^{-1}\}K = \{\{D\}K\}K^{-1} = D$



- Challenge: how to choose K and K^{-1}? 35

Performance

- Asymmetric cryptography 3-5 orders of magnitude slower than symmetric cryptography
- Use asymmetric cryptography to exchange symmetric key; data encrypted using symmetric cryptography:

$$A \rightarrow B: \{K_{AB}\}K_B, \{D\}K_{AB}$$

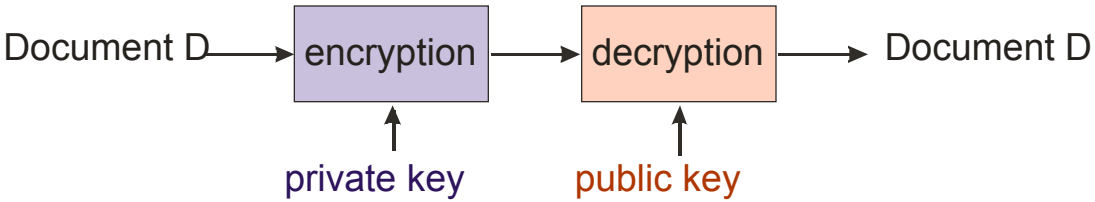
- Asymmetric cryptography has other important uses as well …

Digital Signature (Version 0.1)

- Alice signs document by encrypting it with her own private key

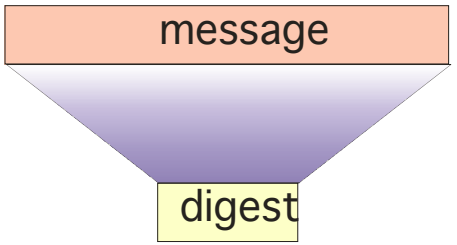
$$A \rightarrow B: \{D\}K_A^{-1}$$

- Bob verifies the signature by decrypting it using A's public key, i.e., compute $D = \{\{D\}K_A^{-1}\}K_A$



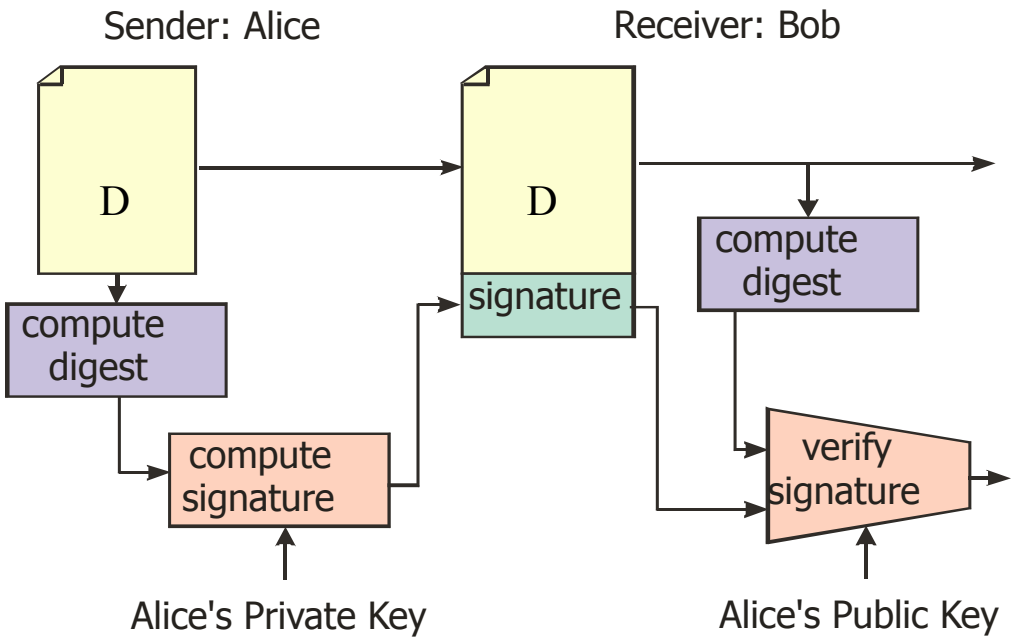
- Two outcomes:
  - digital signature provides integrity and accountability (non-repudiation)
  - Alice is authenticated to Bob. (How?)
- There is another problem -- performance

Cryptographic Hash/ Message Digest



- Hash function maps arbitrary length message D to fixed length digest H(D)
  - –MD5 (128 bit digest) and SHA-1 (160 bit digest) are commonly used
- One-way function: given H(D), can't find D
- Collision-free: infeasible for attacker to generate D and D' such that H(D) = H(D’)

Digital Signature (Version 1.0)



- A → B: D, $\{H(D)\}K_A^{-1}$ → signature
- Bob:
  - Computes hash of message, H(D)
  - "Decrypts" signature: $\{\{H(D)\}K_A^{-1}\}K_A$
  - Verifies H(D) = $\{\{H(D)\}K_A^{-1}\}K_A$