TCP Congestion Control & DNS

IPADS, Shanghai Jiao Tong University

https://www.sjtu.edu.cn

End-to-end Layer

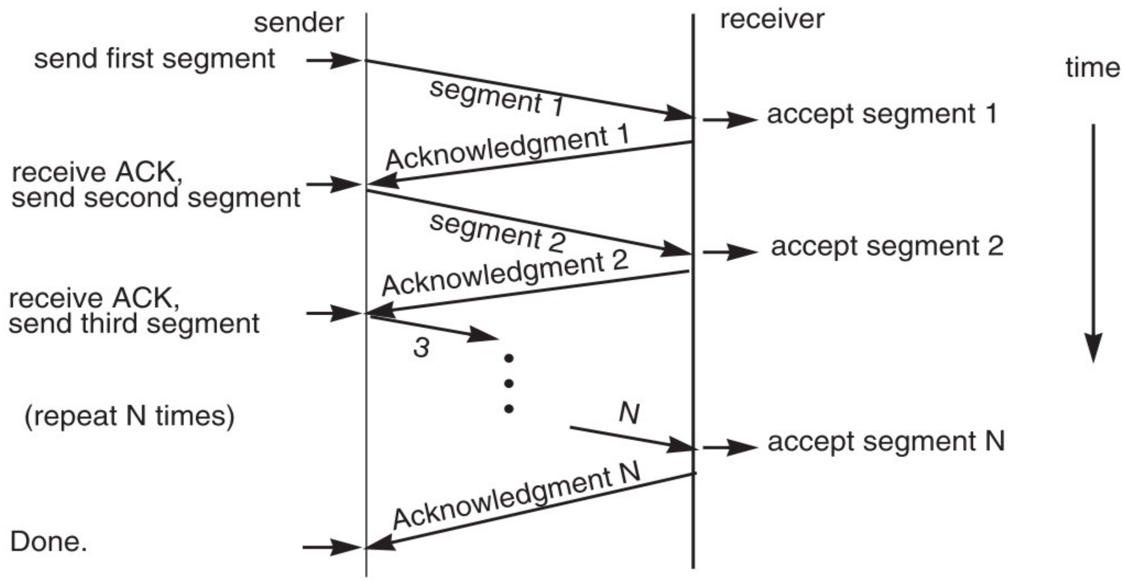
Review: Assurance of End-to-end Protocol

1. Assurance of at-least-once delivery
2. Assurance of at-most-once delivery
3. Assurance of data integrity
4. Assurance of stream order & closing of connections
5. Assurance of jitter control
6. Assurance of authenticity and privacy
7. Assurance of end-to-end performance

7. End-to-end Performance

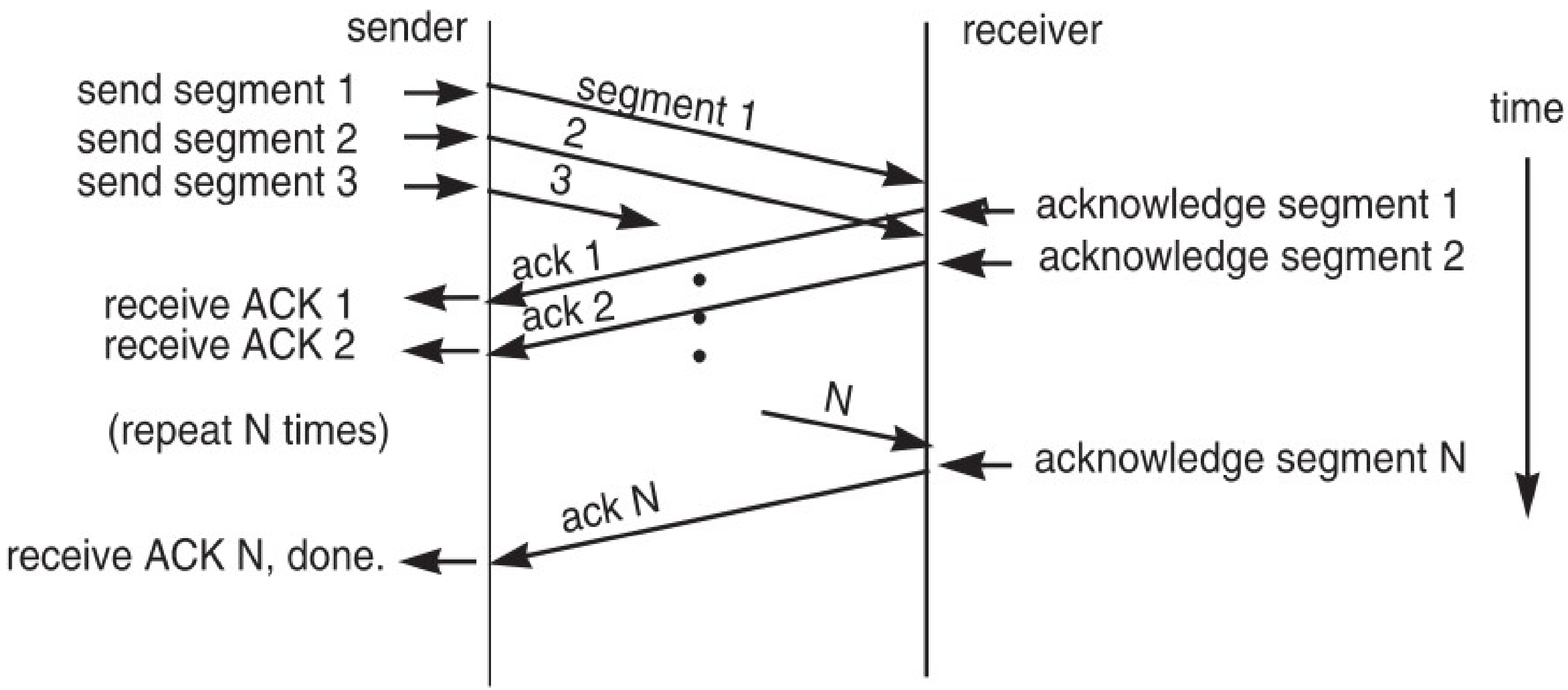
Multi-segment message questions

- Trade-off between complexity and performance
- Lock-step protocol



Overlapping Transmissions

Pipelining technique



Overlapping Transmissions

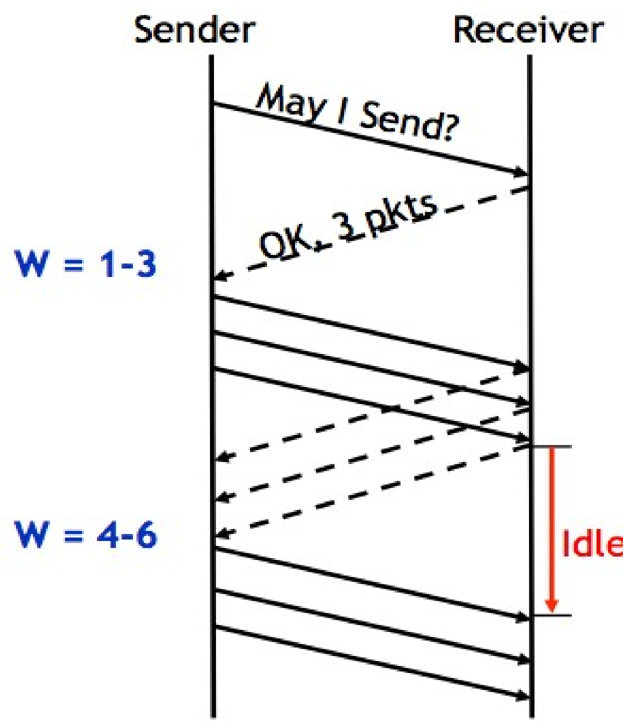
Packets or ACK may be lost

- Sender holds a list of segments sent, check it off when receives ACK
- Set a timer (according to RTT) for last segment

If list of missing ACK is empty, OK

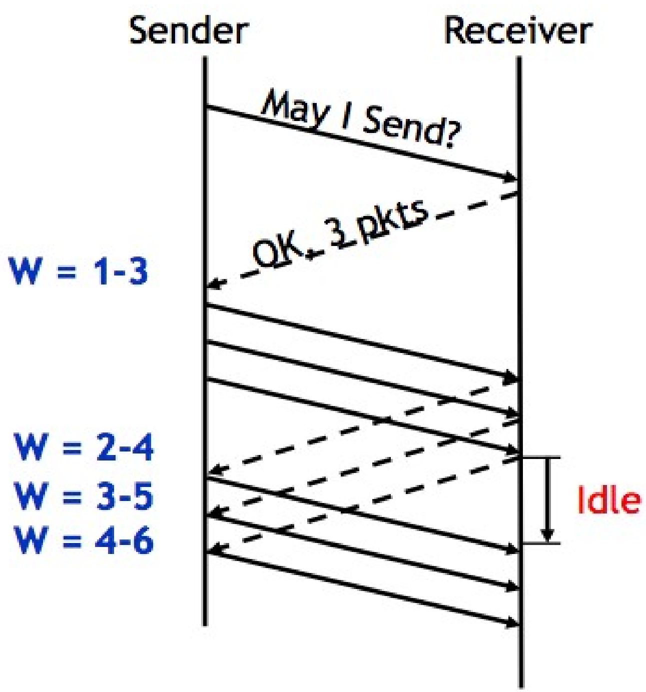
If timer expires, resend packets and another timer

Fixed Window



- Receiver tells the sender a window size
- Sender sends window
- Receiver acks each packet as before
- Window advances when all packets in previous window are acked
  - E.g., packets 4-6 sent, after 1-3 ack'd
- If a packet times out -> resend packets
- Still much idle time

Sliding Window



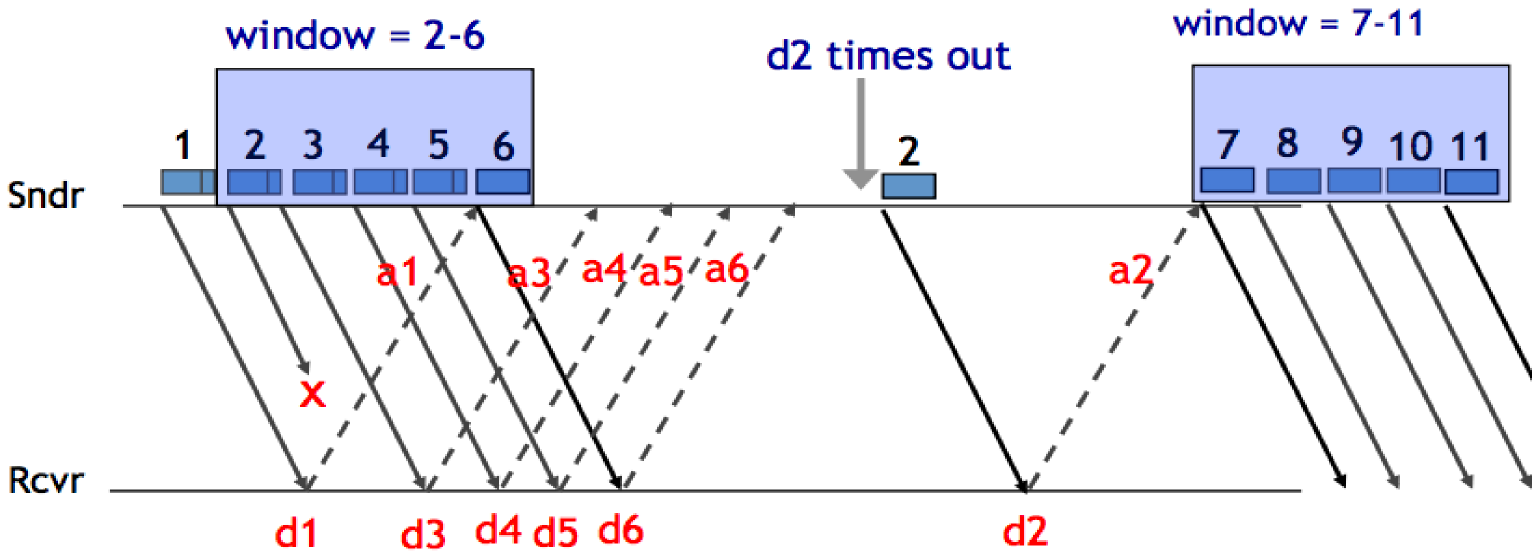
Sender advances the window by 1 for each in-sequence ACK it receives

- Reduces idle periods
- Pipelining idea

But what's the correct value for the window?

- We'll revisit this question
- First, we need to understand windows

Handling Packet Loss



Sender advances the window on arrivals of in-sequence acks

→ Can’t advance on a3’s arrival

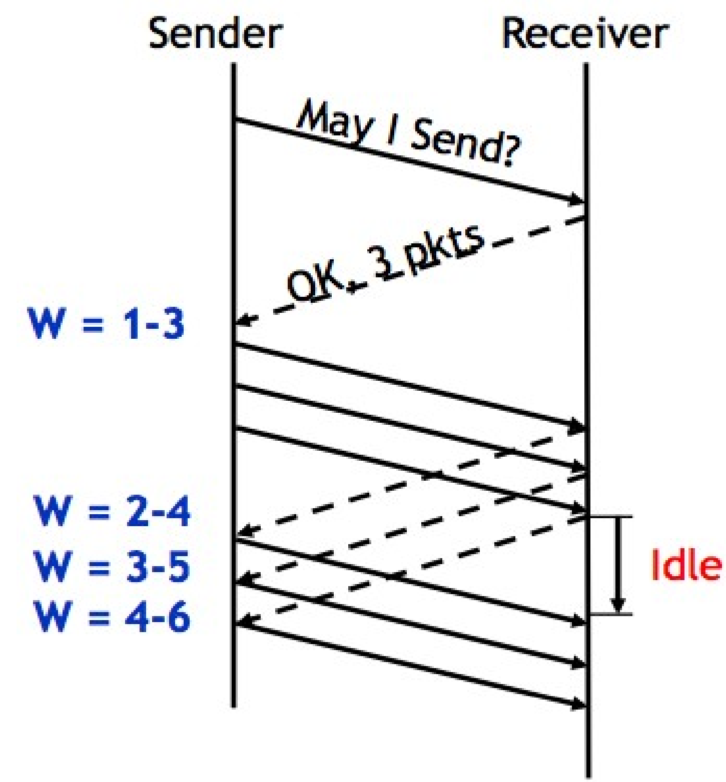
Chose the Right Window Size

If window is too small

- Long idle time
- Underutilized network

If window too large

- Congestion



Sliding Window Size

window size ≥ round-trip time × bottleneck data rate

Sliding window with one segment in size

– Data rate is window size / RTT

Enlarge window size to bottleneck data rate

– Data rate is window size / RTT

Enlarge window size further

– Data rate is still bottleneck

– Larger window makes no sense

- Receive 500 KBps
- Sender 1 MBps
- RTT 70ms
- A segment carries 0.5 KB

- Sliding window size = 35KB (70 segment)

Self-pacing: Sliding Window Size

Although the sender doesn't know the bottleneck, it is sending at exactly that rate

Once sender fills a sliding window, cannot send next data until receive ACK of the oldest data in the window

The receiver cannot generate ACK faster than the network can deliver data elements

RTT estimation still needed

TCP Congestion Control

Congestion

Definition: Too many packets present in (a part of) the network causes packet delay and loss that degrades performance.

Network & End-to-end layers *share the responsibility* for handling congestion

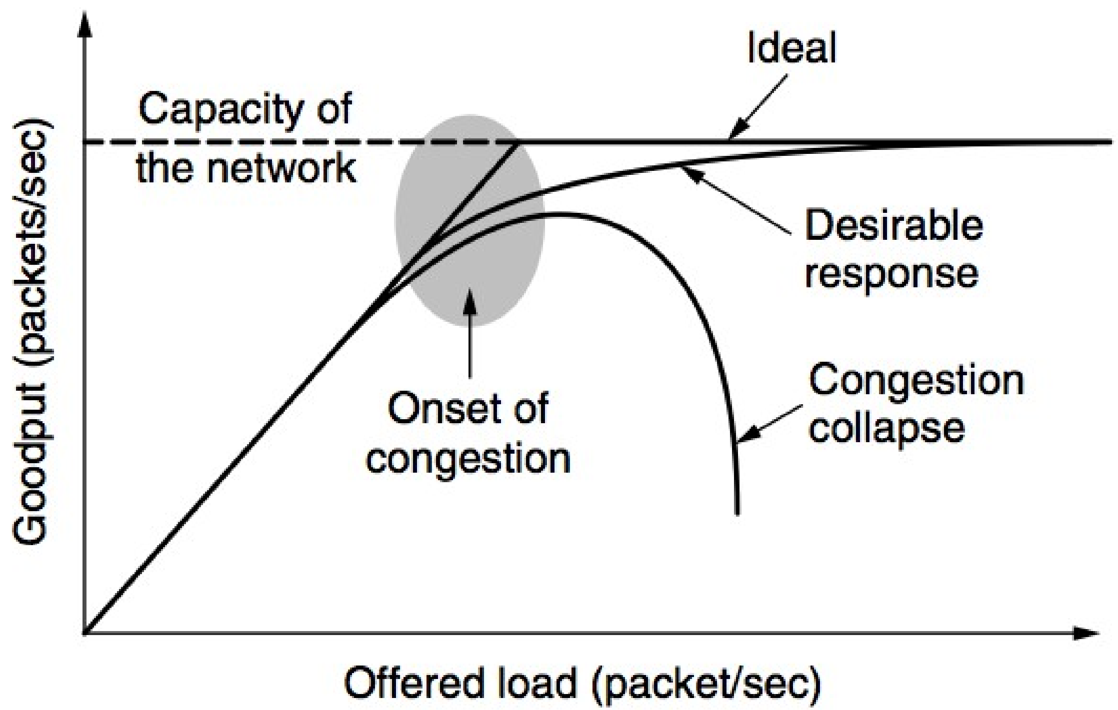
1. Network layer

- Directly experiences the congestion
- Ultimately determine what to do with the excess packets

2. End-to-end layer

- Control to reduce the sending rate, is the most effective way

Network Congestion



Why Congest?

If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up

If there is insufficient memory to hold all of them, packets will be lost

Adding more memory may help up to a point, but

- Nagle (1987) realized that if routers have an infinite amount of memory, congestion gets worse, not better
- This is because by the time packets get to the front of the queue, they have **already timed out** (repeatedly) and duplicates have been sent

Load Shedding: Setting Window Size

For performance:

- window size ≥ round-trip time × bottleneck data rate

For congestion control:

- window size ≤ min(RTT x bottleneck data rate, Receiver buffer)
- Congestion window

2 windows become 1

- to achieve best performance and avoid congestion

Congestion Control

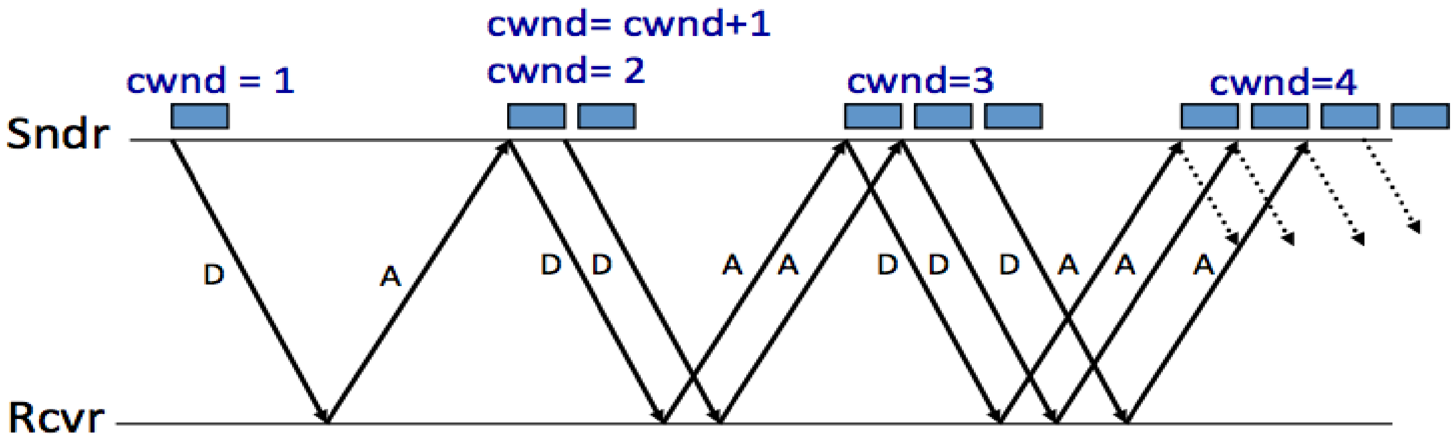
Basic idea:

- Increase congestion window slowly
- If no drops -> no congestion yet
- If a drop occurs -> decrease congestion window quickly

Use the idea in a distributed protocol that achieves:

- Efficiency: i.e., uses the bottleneck capacity efficiently
- Fairness, i.e., senders sharing a bottleneck get equal throughput (if they have demands)

AIMD (Additive Increase, Multiplicative Decrease)



Every RTT:

- No drop: cwnd = cwnd + 1
- A drop: cwnd = cwnd / 2

Problems with AIMD

Increases very slowly at the beginning

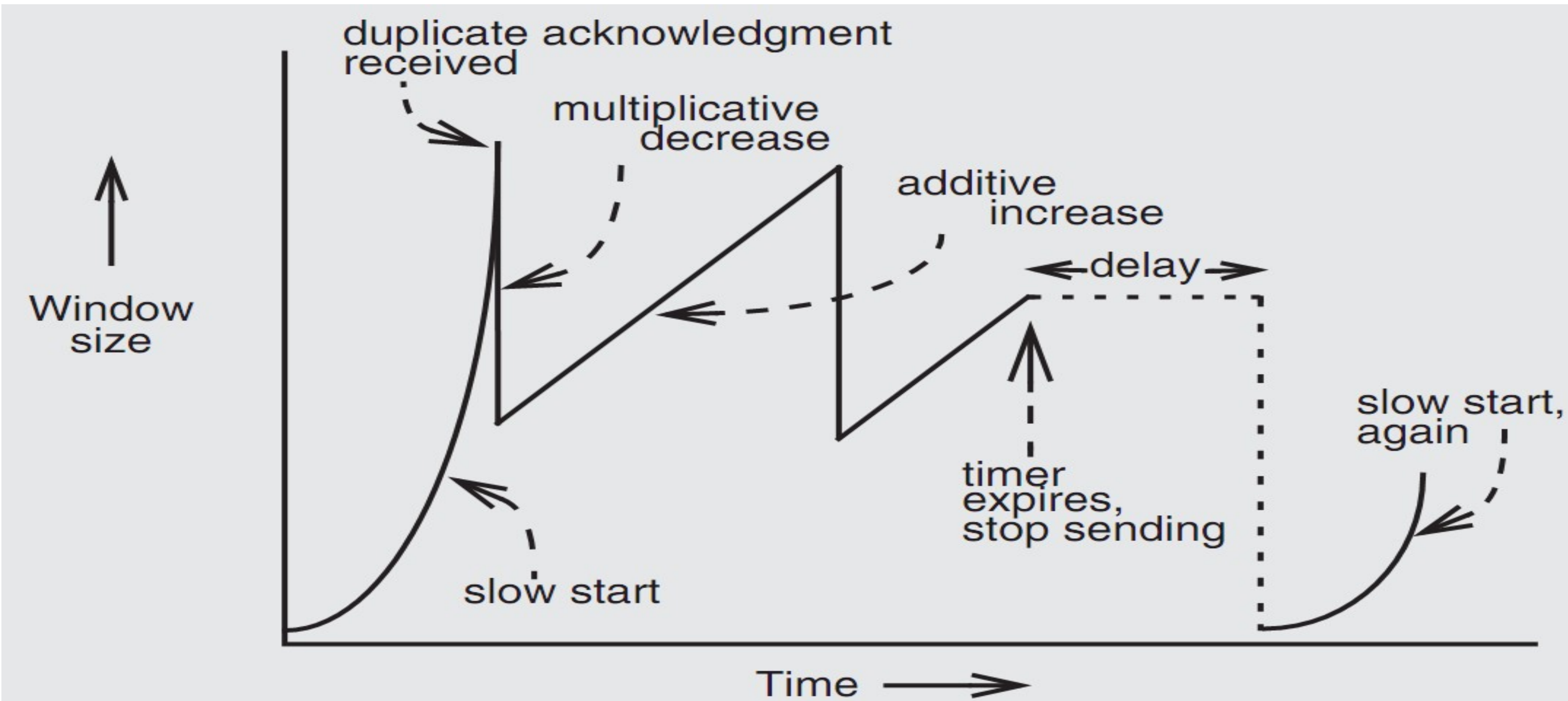
Initial window size is 1

- Probably too small in practice

Solution: do multiplicative increase at the beginning

- *Congestion_window* $_{init}$ = 1
- Initially, do *Congestion_window* ← 2 * *Congestion_window* each RTT until we hit congestion
- Named "slow start" (even though it's exponentially fast!)

Retrofitting TCP



Retrofitting TCP

1. Slow start: one packet at first, then double until

- Sender reaches the window size suggested by the receiver
- All the available data has been dispatched
- Sender detects that a packet it sent has been discarded

2. Duplicate ACK

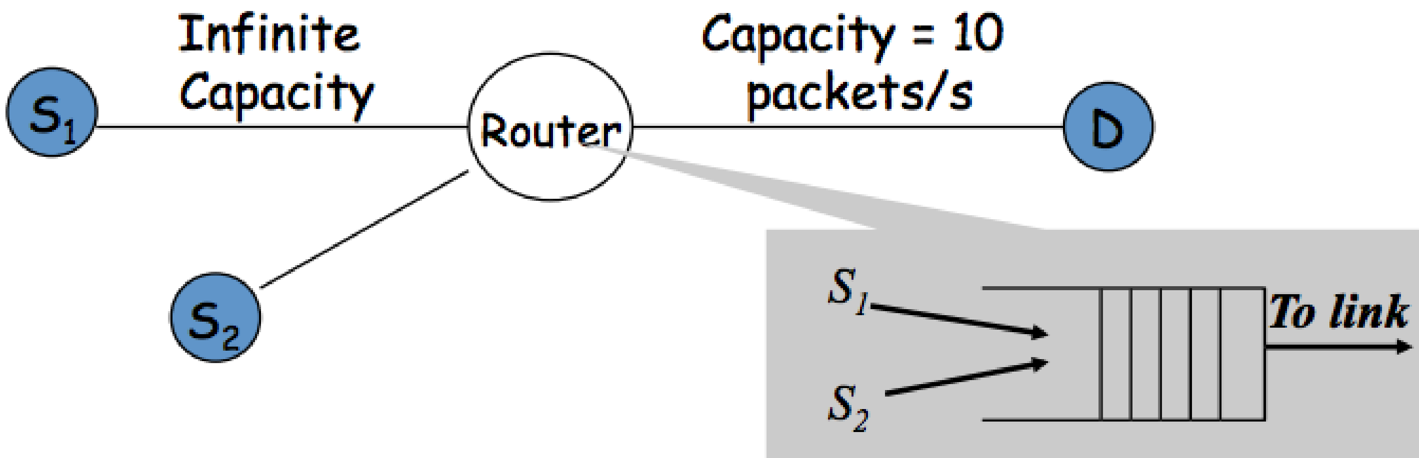
- When receiver gets an out-of-order packet, it sends back a duplicate of latest ACK

3. Equilibrium

- AIMD: Additive increase & multiplicative decrease

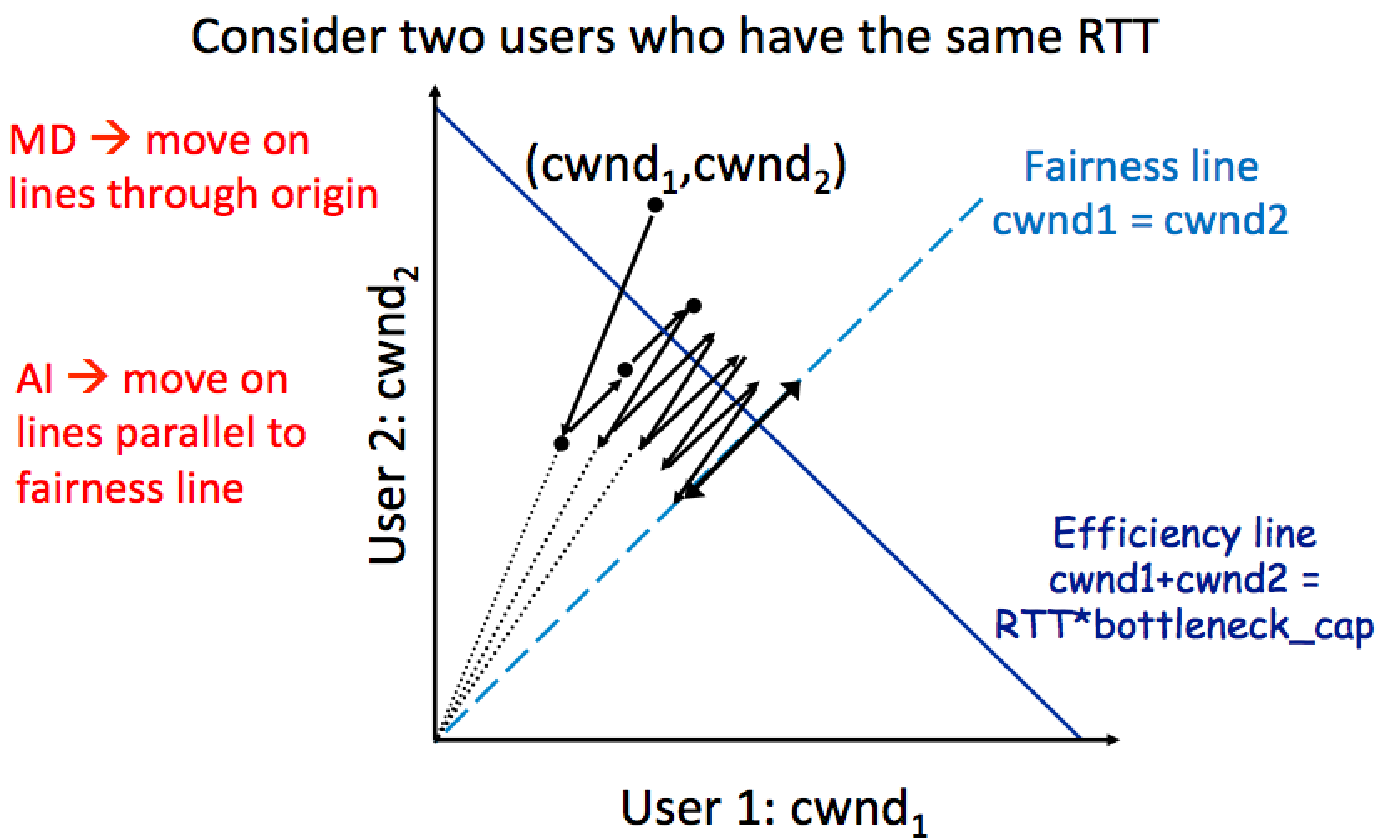
4. Restart, after waiting a short time

Fairness between Links



Bottleneck may be shared

AIMD Leads to Efficiency and Fairness



Q: Why not Additive Decrease

It does not converge to fairness

- from a congested point, (x',y'), reducing each by 1 worsens fairness and takes us away from the "ideal" outcome

Weakness of TCP

If routers have too much buffering, causes long delays

Packet loss is not always caused by congestion

- Consider wireless network: if losing packet, sender may send faster instead

TCP does not perform well in datacenters

- High bandwidth, low delay situations

TCP has a bias against long RTTs

- Throughput inversely proportionally to RTT
- Consider when sending packets really far away vs really close

Assumes cooperating sources, which is not always a good assumption

Summary of Congestion Window

Reliability Using Sliding Window

- Tx Rate = W / RTT

Congestion Control

- W = min(Receiver_buffer, cwnd)
- Congestion window is adapted by the congestion control protocol to ensure efficiency and fairness
- TCP congestion control uses AIMD which provides fairness and efficiency in a distributed way

The Design of DNS

Domain Name Service

DNS: Binding IP and Domain Name

Names: hostname strings

- E.g., www.sjtu.edu.cn

Values: IP addresses

- E.g., 202.120.2.119

Look-up algorithm

- Resolves a hostname to an IP address so that your machine knows where to send packets

IP Address as a Type of Name

An IP address itself is a type of name

- A structured name that is used to locate an object
- Use IP address to identify the server
  - Recall your labs in ICS on socket
- On Internet
  - The router will know where to send a packet with destination IP

Hostname has no such semantic

- A router does not know how to send a packet to "baidu.com"

Why Not Just Using IP Address?

IPs are structured in a particular way for routing

- You cannot chose your IP address as you wish
  - Note: usually an address cannot be picked
- While you can chose your host names

IPs are not user-friendly enough

Questions on DNS

Q: Can a name have multiple values (IP addresses)?

- Yes. This allows a web server to balance its load over multiple machines
- Also allows a client to choice a nearest IP to access

Q: Can a single value have multiple names?

- Yes. This allows server consolidation

Q: Can the value corresponding to a name change?

- Yes. This allows to change the physical machine (with different IP) that stores the data without changing the hostname
- Such changing is hidden to clients

Look-up Algorithm

At first, each machine kept a "hosts.txt" for address binding

- E.g., "r900 202.120.224.83"
- Using table look-up to resolve the binding
- This method cannot scale in Internet

1984, four Berkeley students wrote BIND

- **B**erkeley **I**nternet **N**ame **D**omain
- Still the dominant DNS software in use

Distributing Responsibility

The binding

- Too large to be stored on a single machine
- Thus, the data are stored on many machines
  - As known as "**name servers**"

How to know which name server has a particular binding?

- Solution: structure the hostname
- Names have a hierarchy, e.g., *com*, *net*, *gov*, correspond to "**zones**"
- Zones are mapped to name servers

Name Servers

The root zone

- Maintained by **ICANN**, non-profit

The ".com" zone

- Maintained by **VeriSign**, add for money

The ".sjtu.edu.cn" zone

- Maintained by **SJTU**

DNS Hierarchy (a partial view)

Basic DNS Look-up Algorithm

Example: lookup IP of "ipads.se.sjtu.edu.cn"

Traverse the name hierarchy from the root

- The root will tell us the "cn" name server IP,
- which will tell us the "edu.cn" name server IP,
- which will tell us the "sjtu.edu.cn" name server IP,
- which will tell us the "se.sjtu.edu.cn" name server IP,
- which finally tells us the "ipads.se.sjtu.edu.cn" IP

Such algorithm is called <u>delegation</u>

DNS Lookup

DNS Lookup

DNS Lookup

DNS Lookup

DNS Lookup

Context in DNS

Names in DNS are global (context-free)

- A hostname means the same thing everywhere in DNS

Actually, it should be "ipads.se.sjtu.edu.cn."

- A hostname is a list of domain names concatenated with dots
- The root domain is unnamed, i.e., "." + blank

Fault Tolerant

Each zone can have multiple name servers

- A **delegation** usually contains a list of name servers
- If one name server is down, others can be used

Three Enhancements on Look-up Algorithm

1. The initial DNS request can go to any name server, not just the root server

- Even on your own machine: /etc/hosts
- You can specific name servers in /etc/resolv.conf
- If no record, just return address of the root server

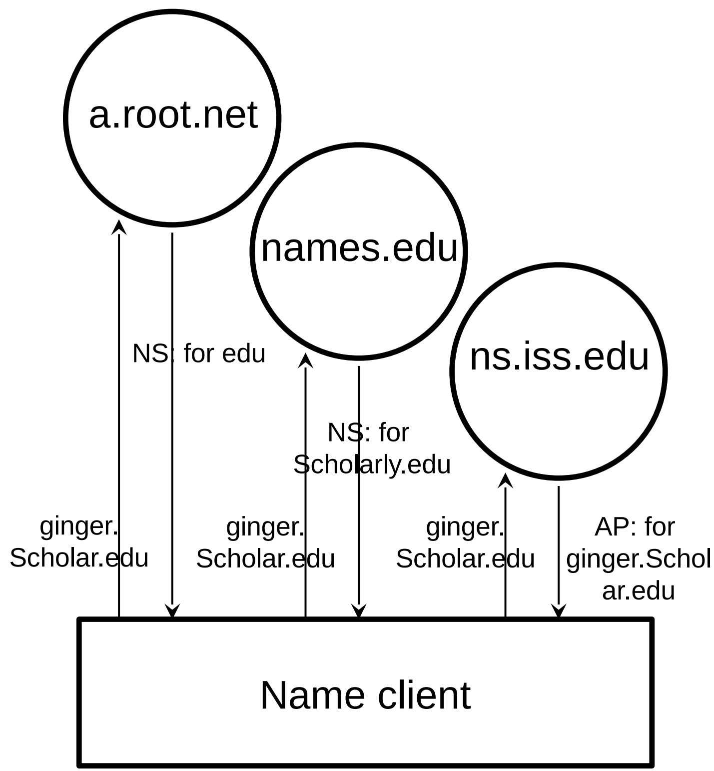
- **Q**: what are the benefits?

Three Enhancements on Look-up Algorithm

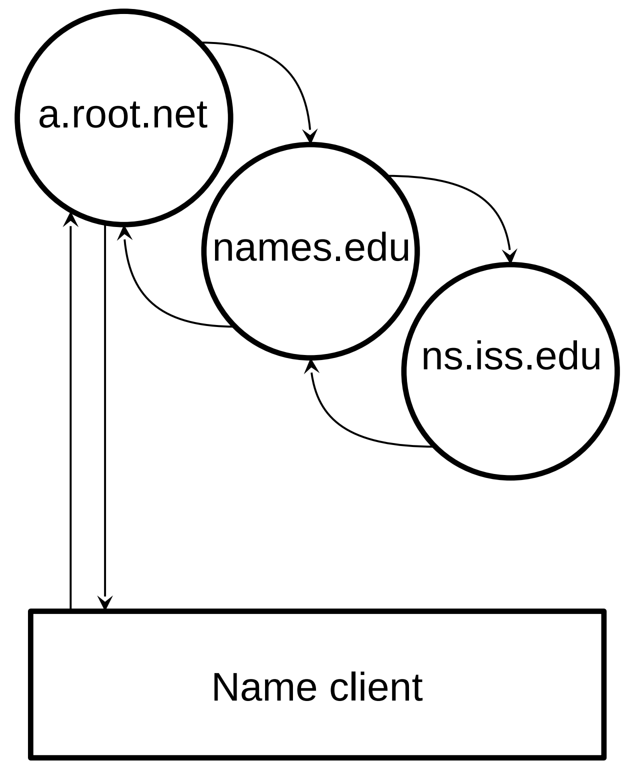
2. Recursion

- A client asks a name server "www.baidu.com"
- The name server **does all the lookup** through the tree and return the IP of Baidu to the client
- Usually, a name server has a better network connection

DNS Request Process



Non-Recursion



Recursion

Three Enhancements on Look-up Algorithm

3. Caching

– DNS clients and name servers keep a cache of names
  - Your browser will not do two look-ups for one address
– Cache has expire time limit
  - Controlled by a time-to-live parameter in the response itself
  - E.g., SJTU sets the TTL of www.sjtu.edu.cn to 24h
– TTL (Time To Live)
  - Long TTL VS. short TTL
  - **Q:** what are the tradeoffs?

Combine These Enhancements

If:

- Many machines at SJTU use the SJTU name server for their initial DNS query, and
- The name server offers recursive querying and caching

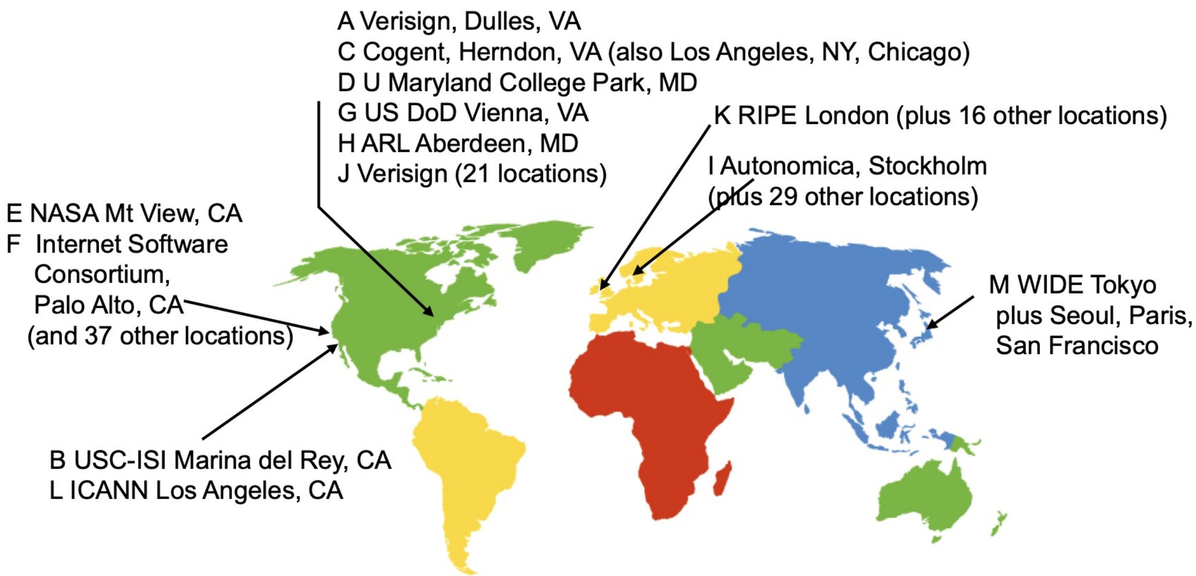
Then:

- The name server's cache will hold many bindings, and
- Performance benefits from this large cache

Other Features of DNS

At least two identical replica servers

- **80 replicas** of the root name server in 2008
- Replicas are placed separated around the world



Other Features of DNS

Organization's name server (e.g., SJTU)

- **Several replicas** in campus
  - To enable communications within the organization
- At least one **out of the campus**
  - To validate the address for outside world

Name Discovery in DNS (at the first place)

A client must discover the name of a nearby name server

- Name discovery broadcast to ISP at first time
- Ask network manager
- Ask by email, Google, etc.

Comparing Hostname & Filename

They are both for more user friendly

- File-name -> inode number
- Host-name -> IP address
- The file-name and host-name are **hierarchical**; inode num and IP addr. are **plane**

They are both not a part of the object

- File-name is not a part of a file (stored in directory)
- Host-name is not a part of a website (stored on name server)

Name and value binding

- File: 1-name -> N-values (no); N-name -> 1-value (yes)
- DNS: 1-name -> N-values (**yes**); N-name -> 1-value (yes)

Behind the DNS Design

Why was DNS designed in this way?

Benefits of Hierarchical Design

Hierarchies delegate responsibility

Each zone is only responsible for a small portion

Hierarchies also limit interaction between modules

- A type of **de-centralization**

Good Points on DNS Design

Global names (assuming same root servers)

- No need to specific a context
- DNS has no trouble generating unique names
- The name can also be user-friendly

Scalable in performance

- Simplicity: look-up is simple and can be done by a PC
- Caching: reduce number of total queries
- Delegation: many name severs handle lookups

Good Points on DNS Design

Scalable in management

- Each zone makes its own policy decision on binding
- Hierarchy is great here

Fault tolerant

- If one name server breaks, other will still work
- Duplicated name server for a same zone

Bad Points on DNS Design

Policy

- Who should control the root zone, .com zone, etc.? Governments?

Significant load on root servers

- Many DNS clients starts by talking to root server
- Many queries for non-existent names, becomes a DoS attack

Security

- How does a client know if the response is correct?
- How does VeriSign know "change Amazon.com IP" is legal?

Naming Scheme

Naming: the glue of modules

Naming in General

- ipads.se.sjtu.edu.cn – hostname
- steven@apple.com - email
- steven – username
- EAX - x86 processor register name
- main() - function name
- WebBrowser - class name
- /courses/cse/index.html - path name (fully-qualified)
- index.html - path name (relative)
- http://ipads.se.sjtu.edu.cn/courses/cse/index.html - URL
- 13918275839- Phone number
- 202.120.40.188 - IP Address

Naming a Disk

File name: /dev/sda1

- As a special type of inode: device inode
- 8,0 as (major, minor)

PCI address (name): 19:00.0

- SCSI storage controller: LSI Logic / Symbios Logic SAS1068E PCI-Express Fusion-MPT SAS (rev 08)

Naming for Modularity

Retrieval: e.g., using URL to get a web page

Sharing: e.g., passing an object reference to a function

- Save space as well: only sending the name, not the object

Hiding: e.g., using a file name without knowing file system

- Can support access control: use an object only if knowing its name
- E.g., Windows has many undocumented API

User-friendly identifiers: e.g., "homework.txt" instead of 0x051DE540

Indirection: e.g., OS can move the location of the file data without notifying the user

- Have you ever defragmented your hard driver?

Addresses as Names

Software uses these names in an obvious way

- E.g., memory addresses

Hardware modules connected to a bus

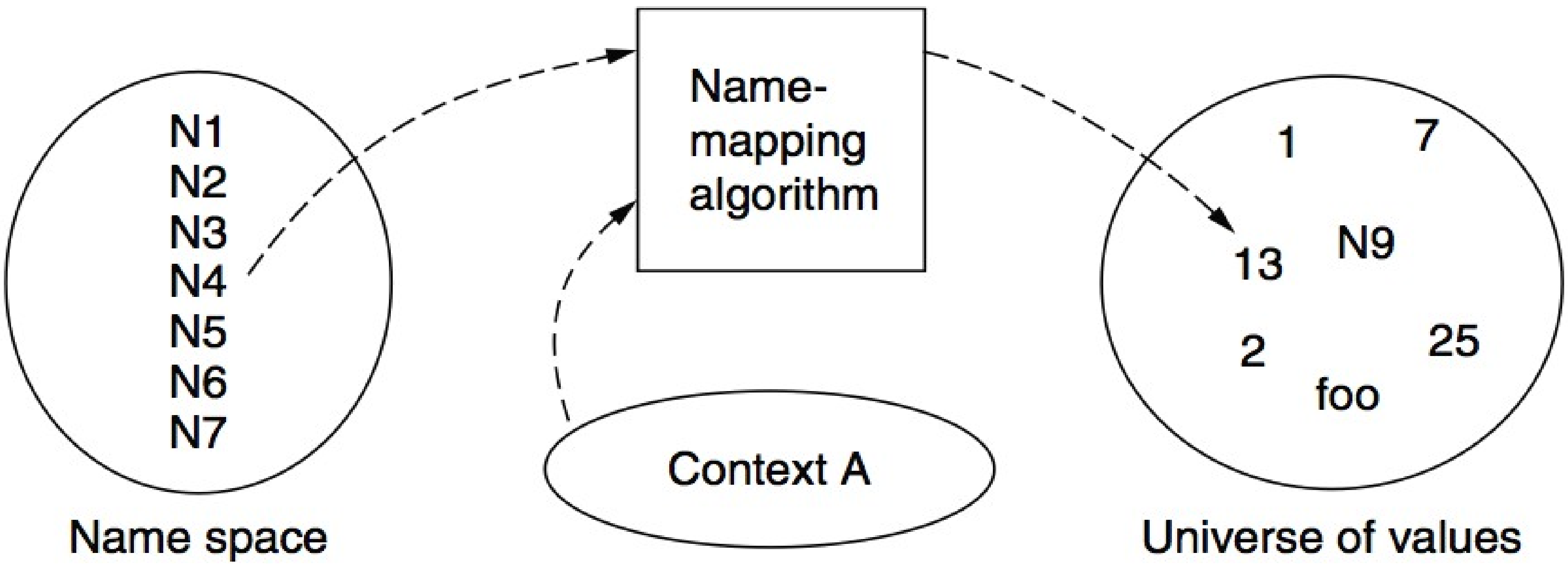
- Use bus addresses (a kind of name) for interconnection

Naming Schemes

A naming schemes contains three parts:

- 1. Set of all possible **names**
  - You cannot use ‘for’ as a variable in C
- 2. Set of all possible **values**
- 3. **Look-up algorithm** to translate a name into a value
  - or a set of values, or “none”

Naming Model



Naming Terminology

Binding – A mapping from a name to value

- Unbind is to delete the mapping
- A name that has a mapping is bound

A name mapping algorithm resolves a name

Naming Context

Type-1: context and name are separated

– E.g., inode number's context is the file system

Type-2: context is part of the name

– E.g., xiayubin@sjtu.edu.cn

Name spaces with only one possible context are called universal name spaces

– Example: credit card number, UUID, email address

Determining Context - 1

Hard code it in the resolver

- Examples: Many universal name spaces work this way

Embedded in name itself

- cse@sjtu.edu.cn:
  - Name = “cse”
  - Context = “sjtu.edu.cn”
- /ipads.se.sjtu.edu.cn/courses/cse/README :
  - Name = “README”
  - Context = “/ipads.se.sjtu.edu.cn/courses/cse”

Determining Context - 2

Taken from environment (Dynamic)

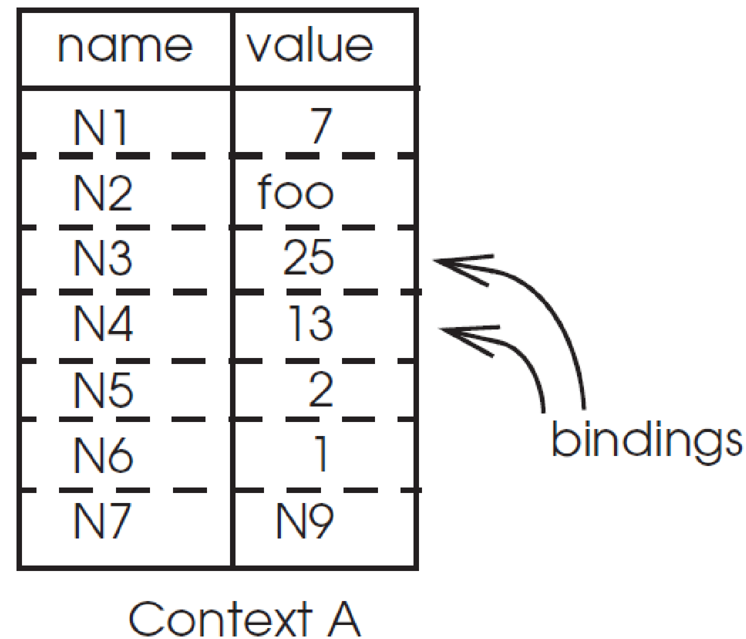
- Unix cmd: "**rm foo**":
  - Name = "foo", context is current dir
  - **Question**: how to find the binary of "rm" command?
- Read memory 0x7c911109:
  - Name = "0x7c911109",
  - Context is thread's address space

Many errors in systems due to using wrong context

Name Mapping Algorithms - 1

Table lookup

- Find name in a table
  - E.g., Phone book
- Context: which table?
  - Implicit VS. explicit
  - Default context



Name Mapping Algorithms - 2

Recursive lookup

- E.g., “/usr/bin/rm”
- First find “usr” in “/”, then find “bin” in “/usr”, then “rm”
- Each look-up process is the same

Multiple lookup

- **Recall**: how to find “rm” without absolute name?
- $PATH
  - E.g., “/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin”
- Look-up in a predefined list of context

Interpreter Naming API

value ← **RESOLVE(*name*, *context*)**

– Return the mapping of *name* in the *context*

status ← **BIND(*name*, *value*, *context*)**

– Establish a *name* to *value mapping* in the *context*

status ← **UNBIND(*name*, *context*)**

– Delete name from *context*

list ← **ENUMERATE(*context*)**

– Return a list of all bindings

result ← **COMPARE(*name1*, *name2*)**

– Check if *name1* and *name2* are equal

FAQ of Naming Scheme - 1

What is the syntax of names?

What are the possible value?

What context is used to resolve names?

Who specifies the context?

Is a particular name global (context-free) or local?

FAQ of Naming Scheme - 2

Does every name have a value?

- Or, can you have “dangling” names?

Can a single name have multiple values?

Does every value have a name?

- Or, can you name everything?

Can a single value have multiple names?

- Or, are there synonyms?

Can the value corresponding to a name change over time?