RW354

Principles of Computer Networking

A.E. Krzesinski and B.A. Bagula
Department of Computer Science
University of Stellenbosch

Virtual Clock: Overview

- *Different approach*
  - *uses reservations rather than feedback*
  - *rate-based rather than window-based*
  - *router-centric rather than host-centric*
- *Idea*
  - *modeled after time-division multiplexing (TDM)*
  - *statistical multiplexing to accommodate bursty traffic*
  - *uses logical (rather than real) time*

Virtual Clock: Defining Rate

- *Explicit flow setup phase*
  - *source indicates its needs*
  - *network grants or denies request*
- *Resource needs*
  - *average rate (*`AR`*)*
  - *average interval (*`AI`*)*
  - *example:* `AR` *= 10 packets per second and* `AI` *= 100ms*
  - *range:* `1/AR` $\leq$ `AI` $\leq$ *total flow duration*
  - *datagram traffic gets it's own flow; granted some capacity*

Virtual Clock: a Queuing Discipline

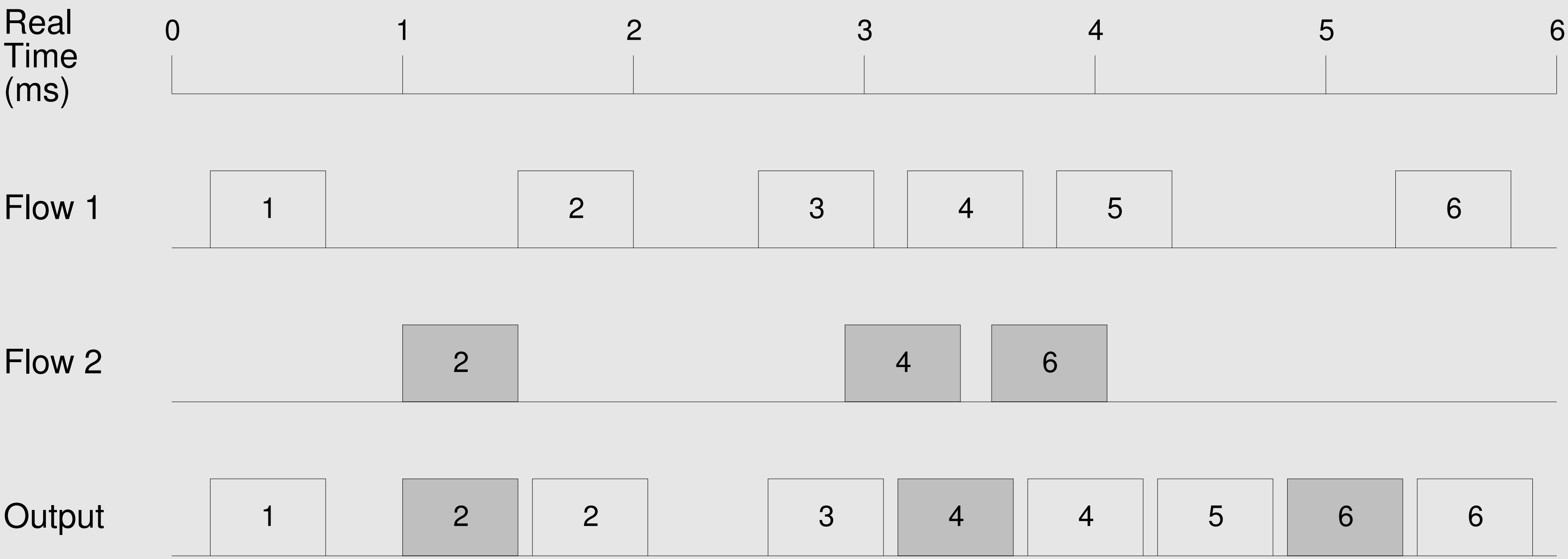
- *Similar to weighted fair queuing*
- *Router variables*
  - $\texttt{VClock}_i$ *for each flow* i
  - *clock tick (*$\texttt{VTick}_i$*) for flow* i
  - $\texttt{VTick}_i \texttt{ = 1/AR}_i$ *(assumes all packets on flow* i *are the same size)*
  - *example: flow* i *has* AR = *200 packets per second, then* $\texttt{VTick}_i$ = *5ms*

Virtual Clock: a Queuing Discipline

- *Router algorithm*
  - *first packet:* `VClock`$_i$ = `RealTime`
  - *each packet thereafter:* `VClock`$_i$ `+=` `VTick`$_i$
  - *timestamp each packet with* `VClock`$_i$
  - *packets queued and serviced in according to timestamps*
  - *when buffer space is full, packet with largest timestamp is dropped*

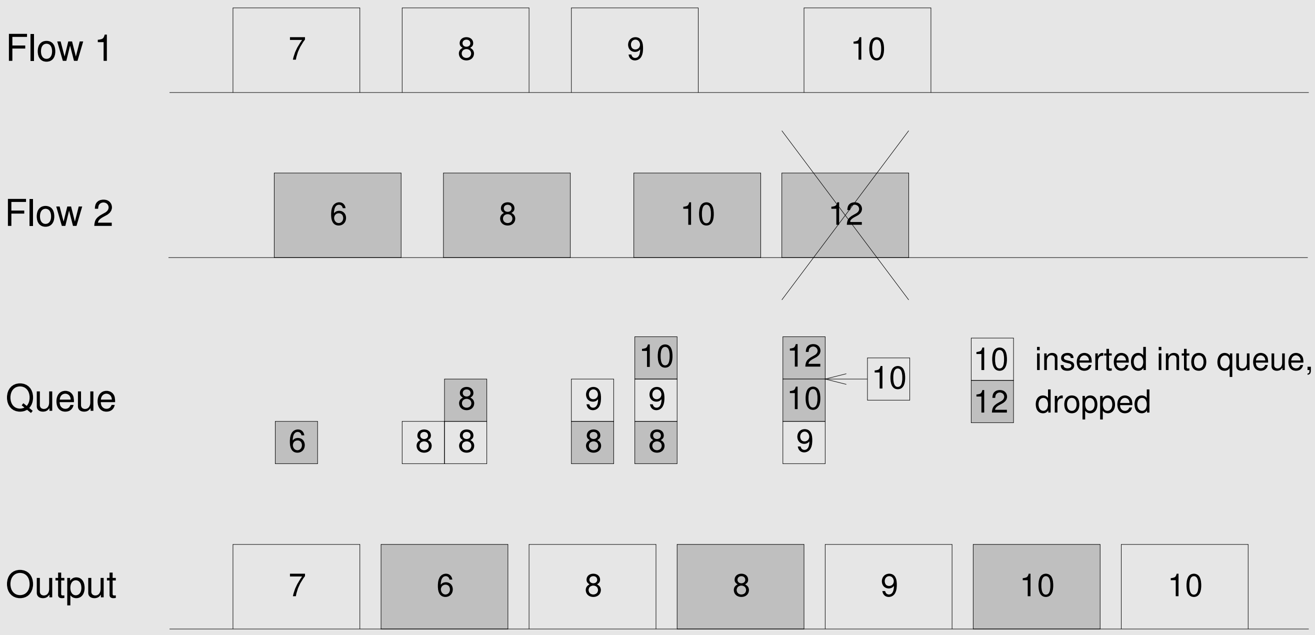
Virtual Clock: a Queuing Discipline

- *Example of interleaved service proportional to* AR
  - *flow 1:* AR = *1000 packets per second*
  - *flow 2:* AR = *500 packets per second*



Virtual Clock: a Queuing Discipline

- *Example of discarding a packet*



Virtual Clock: a Flow Meter

- *Note*
  - *have not yet used* `AI`
  - *possible for flow to "save up" credits and use for burst*
- *Additional mechanism*
  - *when setup flow i, compute* $\texttt{AIR}_i = \texttt{AR}_i \times \texttt{AI}_i$
  - *upon receiving* $\texttt{AIR}_i$ *bytes on flow* i*...*
    - *if* $\texttt{VClock}_i$ - $\texttt{RealTime} > \texttt{Threshold}$, *then send advisory to source*
    - *if* $\texttt{VClock}_i < \texttt{RealTime}$, *then reset* $\texttt{VClock}_i$ *to* `RealTime`
  - *source not allowed to accumulate "unused time" during one* `AI` *period and use it in another*

Virtual Clock: a Flow Meter

- *Last detail*
  - *source can still accumulate credit within an* `AI`
  - *problem: using one variable (*`VClock`*) to control both queuing order and flow monitoring*
  - *solution: introduce auxiliary copy of* `VClock`
  - *upon receiving each packet...*

$\texttt{AuxVClock}_i$ `= MAX(RealTime,` $\texttt{AuxVClock}_i$`)`

$\texttt{VClock}_i$ `=` $\texttt{VClock}_i$ `+` $\texttt{VTick}_i$

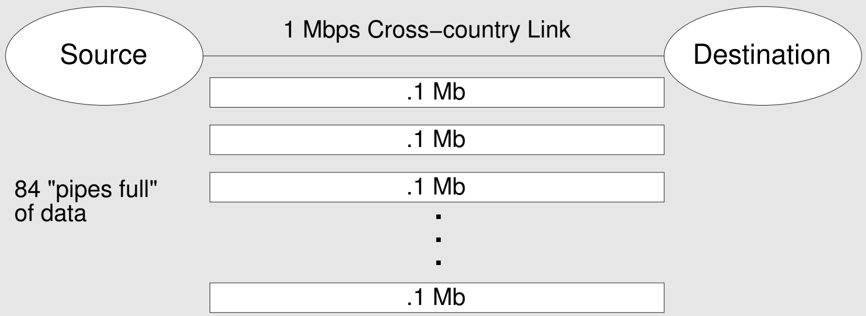
$\texttt{AuxVClock}_i$ `=` $\texttt{AuxVClock}_i$ `+` $\texttt{VTick}_i$

`Stamp packet with current value of` $\texttt{AuxVClock}_i$

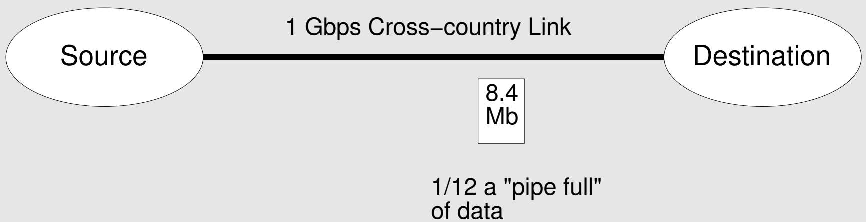
- *Thus*
  - $\texttt{VClock}_i$ *resynchronized with real time once for every* $\texttt{AIR}_i$ *bytes of data on flow* i
  - $\texttt{AuxVClock}_i$ *resynchronized with real time once for every packet on flow* i

High Speed Networks: Overview

- *1Mbps and 1Gbps cross-country links have the same latency (limited by the speed of light)*
- *To transfer a 1MB file takes . . .*
  - *100 RTTs on a 1Mbps link*
  - *doesn't fill a 1Gbps link (12.5MB delay $\times$ bandwidth)*



(a)



(b)

High Speed Networks: Latency/Bandwidth Tradeoff

- *Said another way:*
  - *1MB file is to 1Gbps network what 1KB packet is to 1Mbps network*

High Speed Networks: Latency/Bandwidth Tradeoff

- `Throughput = TransferSize / TransferTime`
  - *if it takes 10ms to transfer 1MB, then the effective throughput is 1MB/10ms = 100MBps = 800Mbps*
- `TransferTime = Latency + 1/Bandwidth` $\times$ `TransferSize`
  - *if network bandwidth is 1Gbps (it takes 1/1Gbps* $\times$ *1MB = 0.8ms to transmit 1MB), an end-to-end transfer that requires 1 RTT of 100ms has a total transfer time of 100.8ms*
  - *effective throughput is 1MB/100.8ms = 79.4Mbps, not 1Gbps*

High Speed Networks: Implications

- *Notes*
  - *transferring a large amount of data helps improve the effective throughput; in the limit, an infinitely large transfer size causes the effective throughput to approach the network bandwidth*
  - *having to endure more than one RTT will hurt the effective throughput for any transfer of finite size, and will be most noticeable for small transfers*

High Speed Networks: Implications

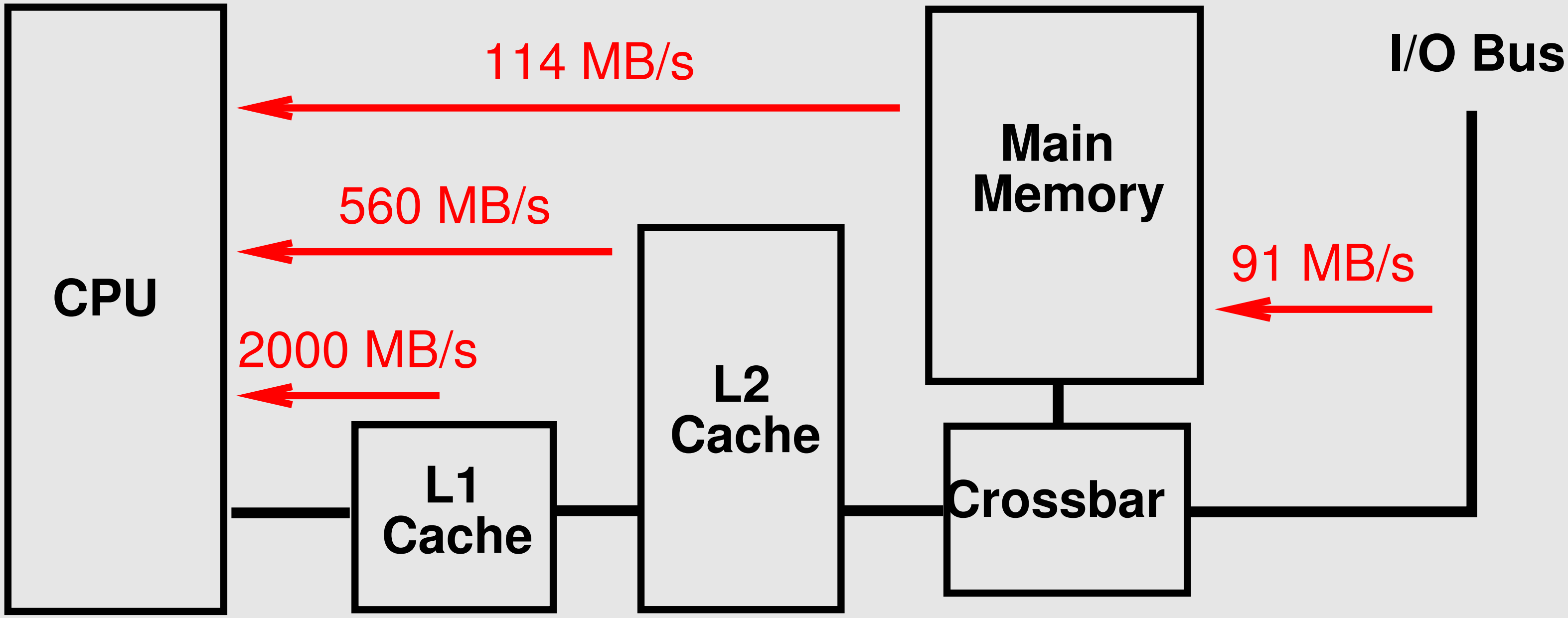
- Congestion control
  - feedback based mechanisms require an RTT to adjust
  - can send 10MB in one 100ms RTT on a 1Gbps network
  - that 10MB might congest a router and lead to massive losses
  - can lose half a delay $\times$bandwidth's of data during slow start
  - reservations work for continuous streams (e.g., video), but require an extra RTT for bulk transfers

High Speed Networks: Implications

- *Retransmissions*
  - *retransmitting a packet costs 1 RTT*
  - *dropping even one packet (cell) halves effective bandwidth*
  - *retransmission also implies buffering at the sender*
  - *possible solution: forward error correction (FEC)*
- *Trading bandwidth for latency*
  - *each RTT is precious*
  - *willing to “waste” bandwidth to save latency*
  - *example: prefetching*

Host Memory Bottleneck: Overview

- *Issue*
  - *turning host-to-host bandwidth into application-to-application bandwidth*
  - *have to deliver data across I/O and memory buses into cache and registers*



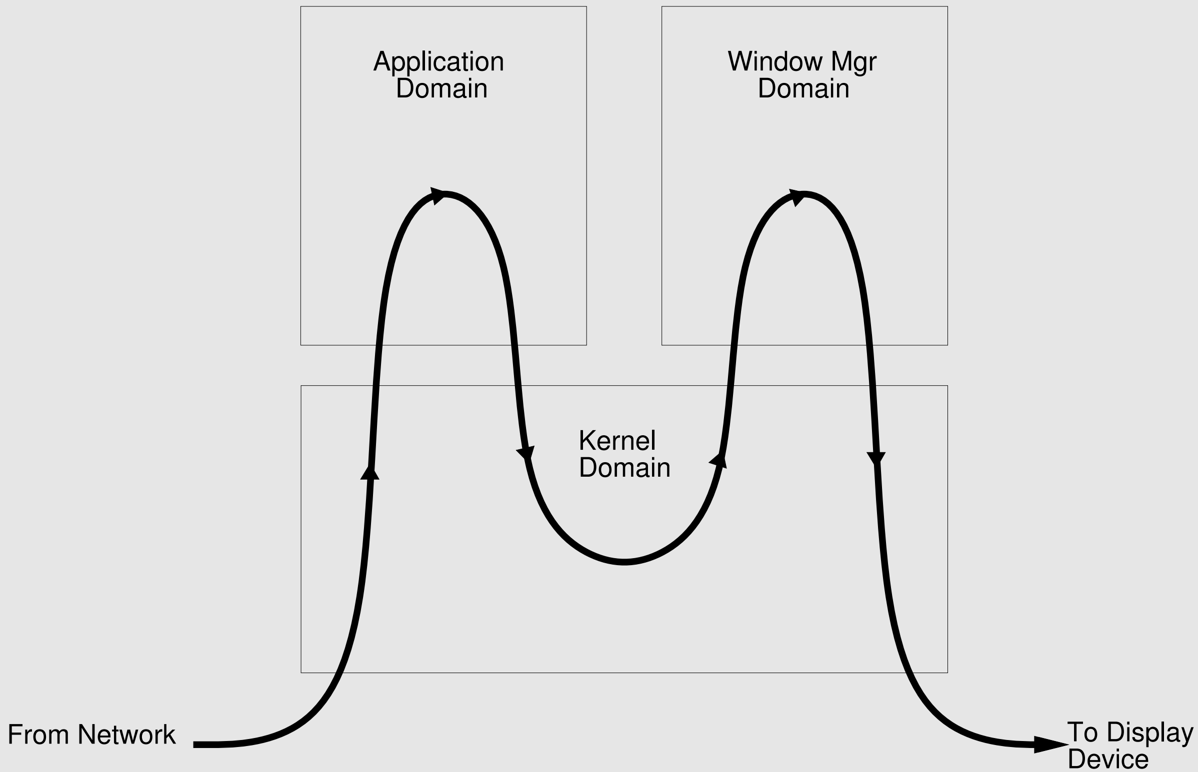
Host Memory Bottleneck: Memory bandwidth

- I/O bus must keep up with network speed (currently does for STS-12, assuming peak rate is achievable)
- 114MBps (measured number) is only slightly faster than I/O bus; can't afford to go across memory bus twice
- caches are of questionable value (rather small)
- lots of reason to access buffers
  - user/kernel boundary
  - certain protocols (reassembly, checksumming)
  - network device and its driver

Same latency/bandwidth problems as high-speed networks

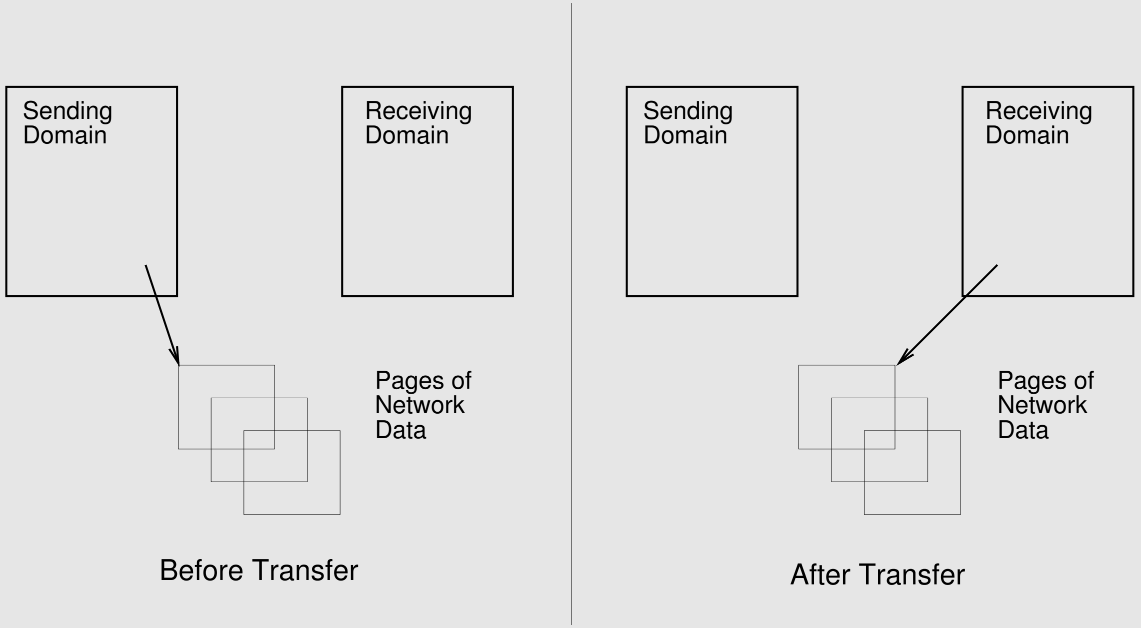
Techniques for Avoiding Data Transfers

- *Device/memory transfers*
  - *mechanism: DMA vs PIO*
  - *transfer size: cell vs packet*
- *Cross-domain transfers*



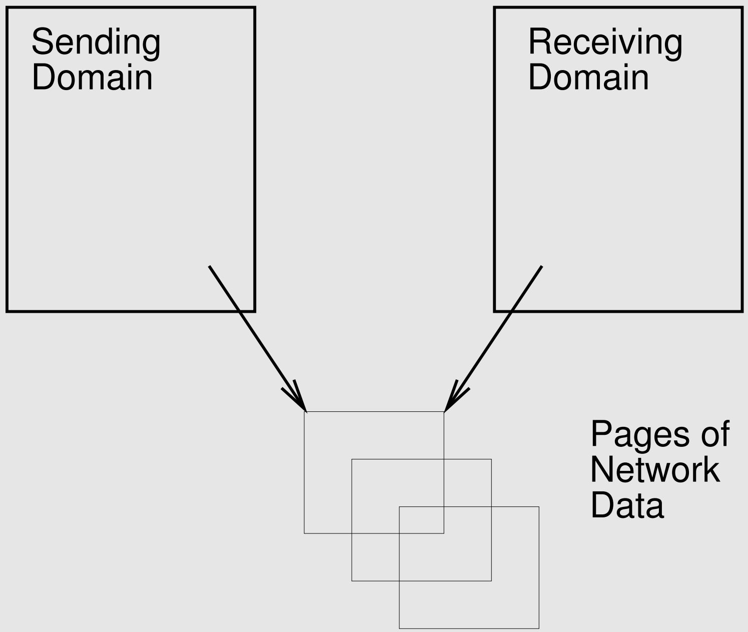
Techniques for Avoiding Data Transfers

- *page remapping*



Host Memory Bottleneck: Data manipulations

- *shared memory*



- *dynamic page sharing (fbufs)*
- *compression, encryption, checksum, presentation formatting*

Host Memory Bottleneck: Data manipulations

- *involves memory loads and stores*
- *integrated layer processing (ILP)*

```
for( i = 0; i < 10000; i++ )
    msgData[i]++;                    /* LOAD, ADD, STORE */

for( i = 0; i < 10000; i++ )
    msgData[i] = ~msgData[i];        /* LOAD, COMPLEMENT, STORE */
```

(a) Two For–Loops

```
for( i = 0; i < 10000; i++ ){
    temp = msgData[i];               /* LOAD */
    temp++;                          /* ADD */
    temp = ~temp;                    /* COMPLEMENT */
    msgData[i] = temp;               /* STORE */
}
```

(b) Integrated For–Loops

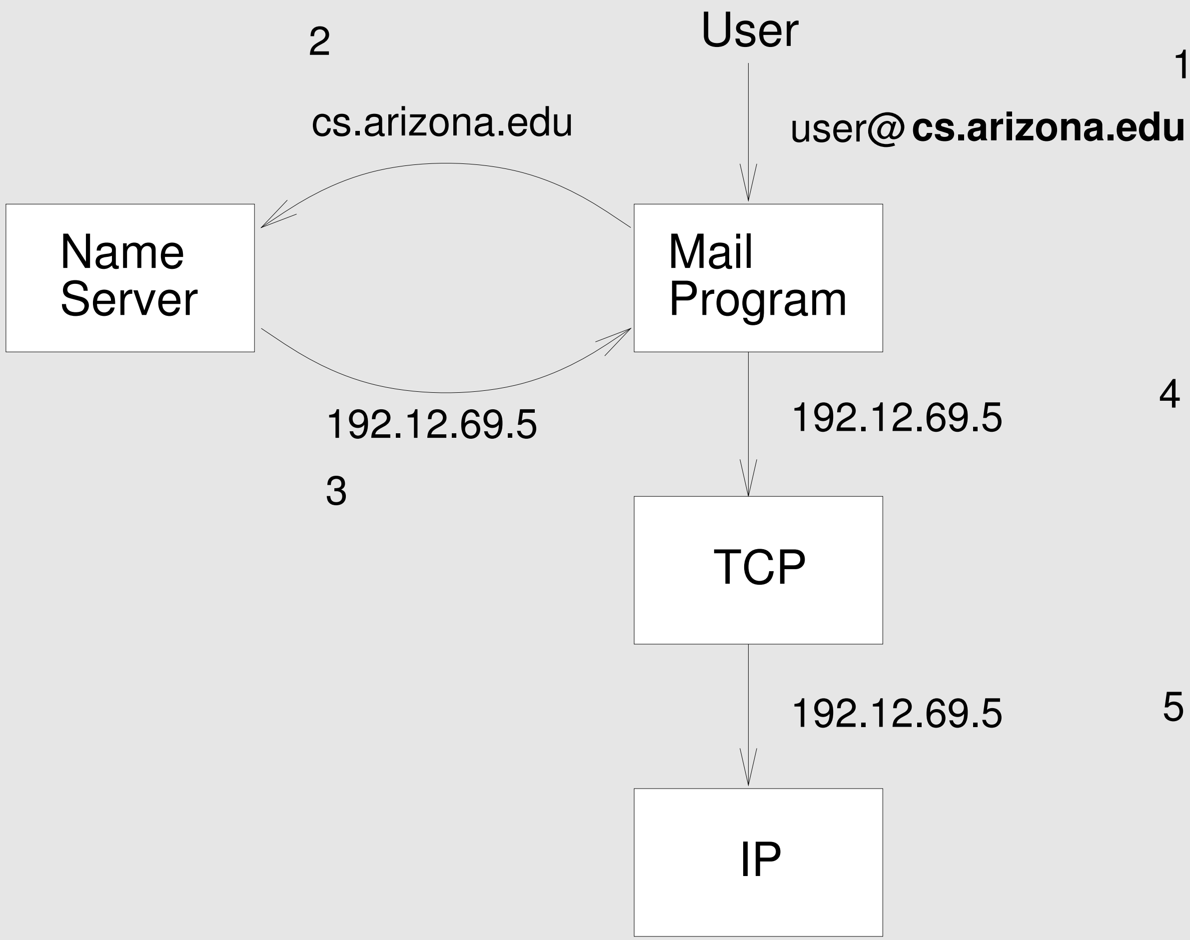
Host Memory Bottleneck: Data manipulations

- *API design*
  - *write: application reuses buffer*
  - *read: application specifies buffer*

DNS: Overview

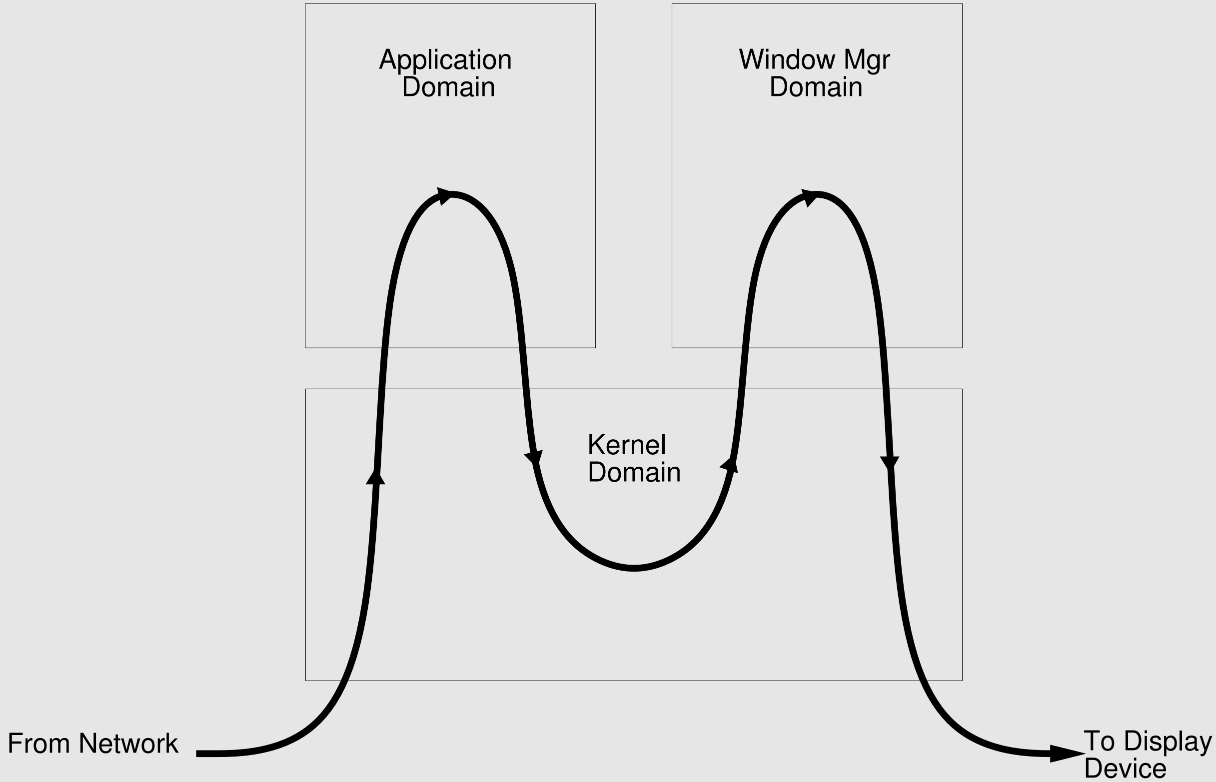
- *Names versus Addresses*
  - *names are variable length, mnemonic, easy for humans to remember*
  - *addresses are fixed length, tied to routing, and easy for computers to process*
- *Name Space*
  - *defines set of possible names*
  - *flat versus hierarchical*
  - *consists of a set of name to value bindings*

DNS: Overview



DNS: Domain Hierarchy

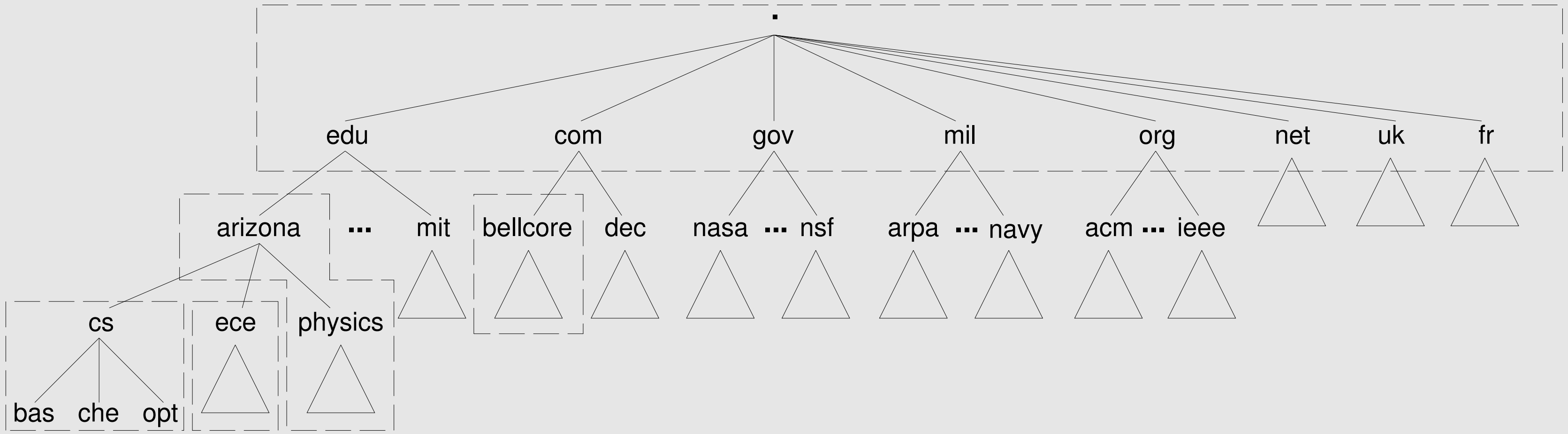
Example hierarchy



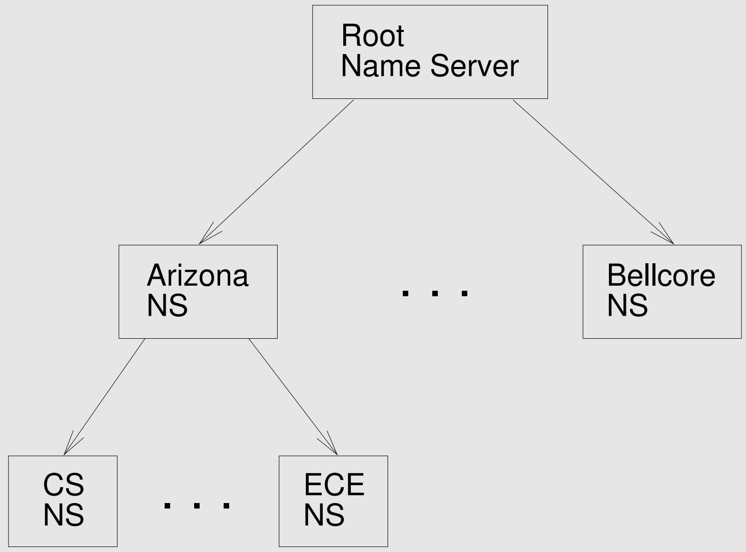
Example name: `cheltenham.cs.arizona.edu`

DNS: Name Servers

Partition hierarchy into zones



Each zone implemented by two or more name servers



DNS: Resource Records

- *Each name server maintains a collection of resource records*

  ⟨`Name, Value, Type, Class, TTL`⟩

- `Name/Value`*: not necessarily host names to IP addresses*
- `Type`
  - `NS`*: the* `Value` *field gives the domain name for a host running a name server that knows how to resolve names within the specified domain.*
  - `CNAME`*: the* `Value` *field gives the canonical name for a particular host; it is used to define aliases.*
  - `MX`*: the* `Value` *field gives the domain name for a host running a mail server that accepts messages for the specified domain.*

DNS: Resource Records

- `Class`: *allow other entities to define types*
- `TTL`: *how long the resource record is valid*

DNS: Example

Root server:

⟨`arizona.edu,telcom.arizona.edu,NS,IN`⟩

⟨`telcom.arizona.edu128.196.128.233AIN`⟩

⟨`bellcore.comthumper.bellcore.comNSIN`⟩

⟨`thumper.bellcore.com128.96.32.20AIN`⟩

`...`

DNS: Example

Arizona server:

$\langle$`cs.arizona.eduoptima.cs.arizona.eduNSIN`$\rangle$

$\langle$`optima.cs.arizona.edu192.12.69.5AIN`$\rangle$

$\langle$`ece.arizona.eduhelios.ece.arizona.eduNSIN`$\rangle$

$\langle$`helios.ece.arizona.edu128.196.28.166AIN`$\rangle$

$\langle$`jupiter.physics.arizona.edu128.196.4.1AIN`$\rangle$

$\langle$`saturn.physics.arizona.edu128.196.4.2AIN`$\rangle$

$\langle$`mars.physics.arizona.edu128.196.4.3AIN`$\rangle$

$\langle$`venus.physics.arizona.edu128.196.4.4AIN`$\rangle$

`...`

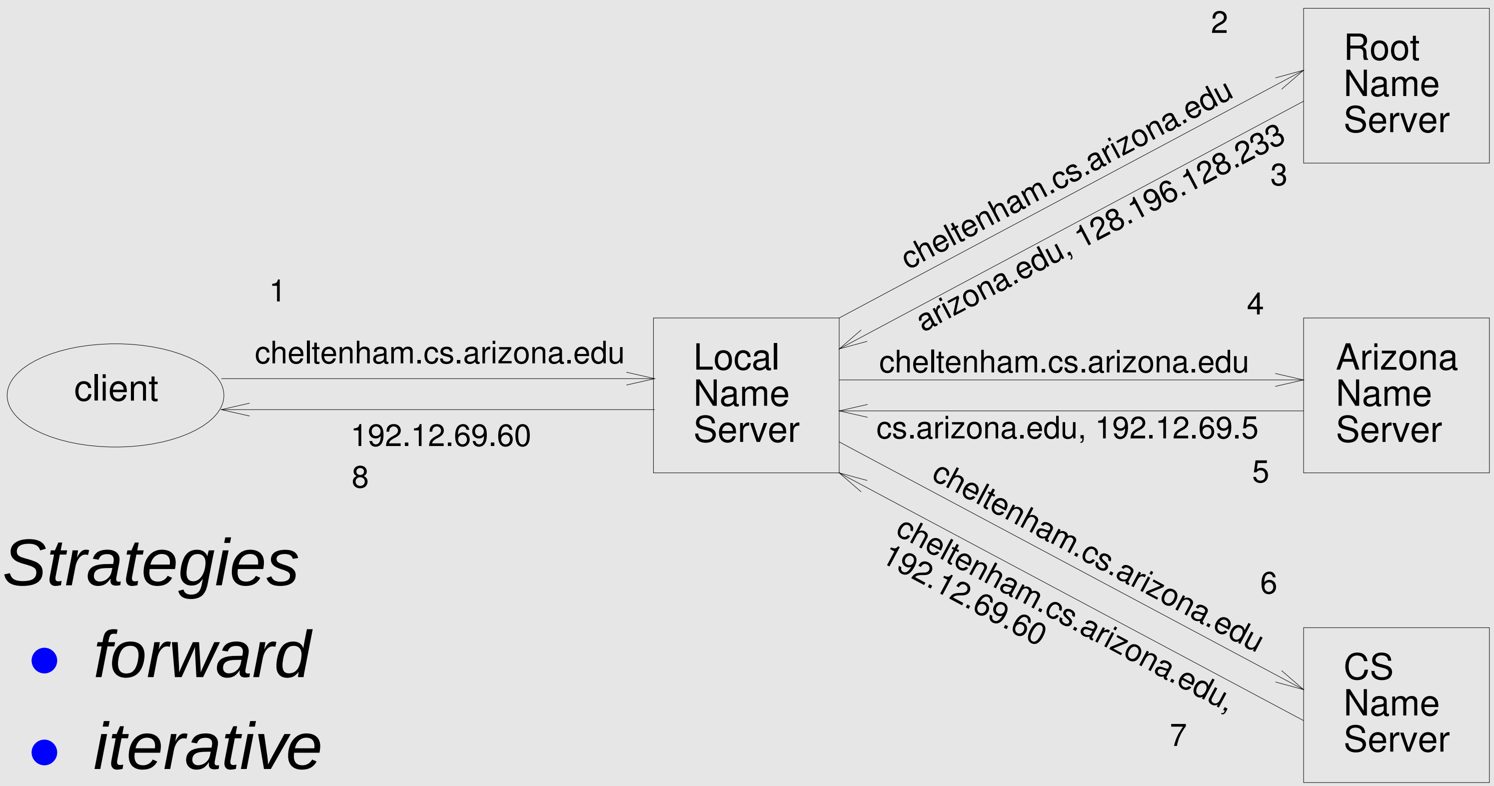
DNS: Example

CS server:

⟨`cs.arizona.eduoptima.cs.arizona.eduMXIN`⟩
⟨`cheltenham.cs.arizona.edu192.12.69.60AIN`⟩
⟨`che.cs.arizona.edu, cheltenham.cs.arizona.edu, CNAME, IN`⟩
⟨`optima.cs.arizona.edu192.12.69.5AIN`⟩
⟨`opt.cs.arizona.eduoptima.cs.arizona.eduCNAMEIN`⟩
⟨`baskerville.cs.arizona.edu192.12.69.35AIN`⟩
⟨`bas.cs.arizona.edu, baskerville.cs.arizona.edu, CNAME, IN`⟩

`.....`

DNS: Name Resolution



- *Strategies*
  - *forward*
  - *iterative*
  - *recursive*
- *Local server*
  - *need to know root at only one place (not each host)*
  - *site-wide cache*