University of Houston

Introduction to Computer Networks

COSC 6377

Final Review

Author
K.M. Hourani

Based on Notes By
Dr. Omprakash Gnawali

May 3, 2021

Contents

1 End-To-End Arguments in System Design

- Design principles that help guide placement of functions among modules of distributed computer systems
- end-to-end argument
  - suggests that functions placed at low levels are redundant or of little value compared to cost
  - "can only be completely and correctly immplemented with knowledge and help of application standing at end points of communication"
- careful file transfer
  - move file from A to B without damage
  - can reinforce all steps by repetition
    - may be uneconomical
  - alternate approach to "check and retry"
    - send checksum
    - if failure probability low, will probably work on first try
  - in order to achieve, program must
    - supply file-transfer specific, end-to-end reliability guarantee
      - checksum to detect failures
      - retry/commit plan
  - thus, even if data communication system is reliable, burder on application is not reduced
- performance tradeoff
  - if too unreliable, performance suffers because of frequent retries
  - if internal reliability added, performance suffers because of redundant data (e.g. checksums)
  - "proper" tradeoff requires careful thought
- similar arguments for
  - delivery guarantees
  - secure transmission of data
  - duplicate message suppression
  - FIFO message delivery
  - transaction management
- must analyze the specific application requirements
- in the end, sort of an "Occam's razor"

2 Dynamics of Random Early Detection

- RED gateway drops packs with dynamically computed probability
  - when average number of packets queued exceeds threshold $\mathtt{min}_{\mathtt{th}}$
  - FCFS scheduling
  - percentage dropped from $\mathtt{connection}_i$ with input rate λ_i

$$\frac{\lambda_i p}{\sum \lambda_i p} = \frac{\lambda}{\sum \lambda_i}$$

  - output rate

$$\frac{\lambda_i(1-p)}{\sum \lambda_i(1-p)} = \frac{\lambda}{\sum \lambda_i}$$

- RED drops packets in proportion to each connection's output usage

- if congestion is persistent, average queue length is above $\texttt{min}_{\texttt{th}}$
  - non-zero minimu drop probability regardless of bandwidth useage
- unfair link sharing
  1. bias against fragile connections
  2. accepting packet from one connection causes higher drop probability for future packets from other connections, even if they consume less bandwidth
  3. non-adaptive connection can force RED to drop packets at high rate from all connections
- Flow Random Early Drop (FRED)
  - modified version of RED
  - behaves like RED with $\texttt{min}_q$ and $\texttt{max}_q$ goals
    * minimum and maximum number of packets each flow allowed to buffer
  - flows with fewer than `avgcq` packets queued are favored over flows with more
  - maintains count of buffered packets `qlen` for each flow
  - maintains variable strike for each flow
    * counts the number of times flow has failed to respond to congestion notification
    * penalizes flows with high strike values
- simulations
  - RED
    * does not provide fair bandwidth sharing
  - FRED
    * provides selective dropping based on per-active-flow buffer counts
    * compatible with existing FIFO queueing architectures
    * often fairer than RED when connections have different RTTs and window sizes
    * protects adaptive flows from non-adaptive flows by enforcing dynamic per-flow queueing limits

3 Revisiting IP Multicast

- New implementation of multicast called Free Riding Multicast (FRM)
  - avoids need of distributed multicast route computation by leveraging unicast routes
  - participation and use is effected via same channel as unicast, BGP (familiar framework)
- moves cost from protocol to router internals
- areas that would benefit from multicast
  - MMORPGs
  - internet TV
  - file-sharing, RSS, software updates, video conferencing, grids
- application layer solutions less constrained by concerns of ISPs
  - difficult to scale
- network layer solutions scale by augmenting existing global ecosystem

- appealing for general-purpose services

- Design
  - group membership discovery
    - group addresses encoded using bloom filter
      - values are hashed, corresponding bits are set in filter
      - membership checked by checking corresponding bits
      - false positives but no false negatives
      - can receive traffic not interested in
      - either drop such traffic, or can inform upstream to stop forwarding such traffic
    - false positive rate of $\min(1, f/(A-G))$, where G is number of groups, A size of address space, and f is number of allowed filters
    - since false positive can only be triggered by one of $A-G$ addresses
    - adds memory requirement
      - marginal in terms of cost
  - multicast packet forwarding
    - forwarding at R_s
      -
  - Evaluation
    -

4 Reverse Traceroute

- traceroute measures sequence of routers from source to destination, with RTT at each hop
- attempt to build a reverse path tool equivalent to traceroute
- source requests a path from system, coordinates probes from source and set of distributed vantage points
  - issue traceroutes to source, yielding atlas of paths to it
  - use this limited view to bootstrap measurement of desired path
- once path from destination reaches hop in the atlas, use atlas to derive remainder of path
- identify reverse hops with IP options
  - `RR-Ping`($S \to D$)
    - S issues ICMP Echo Request to probe to D with RR option
    - if RR slots remain on response, routers record some of route
    - allows limited measurement of reverse path as long as destination is fewer than 9 hops from S
  - `TS-Query-Ping`($S \to D \mid D, R$)
    - S issues ICMP ping probe to D with timestamp query
    - R records timestamp only if encountered by probe after D has stamped packet
    - if S receives timestamp for R, then knows R appears on reverse path
- incrementally build paths