Network Layer Enhancements

EECS 122: Lecture 14

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley

Today

- We have studied the network layer mechanisms that enable “best effort service”
- Today’s focus: How to make best effort better!
  - How are virtual circuits established?
    - Why do they help in improving performance?
  - What does it mean to treat the packets of a network “fairly”
    - Max Min Fairness
  - What network layer mechanisms improve performance?
    - Scheduling
    - Policing
Network Service Models

Q: What service model for “channel” transporting datagrams from sender to rcvr?

Example services for individual datagrams:
- Guaranteed delivery
- Guaranteed delivery with less than 40 msec delay

Example services for a flow of datagrams:
- In-order datagram delivery
- Guaranteed minimum bandwidth to flow
- Jitter Control: Restrictions on changes in inter-packet spacing

Need Virtual Circuits to Provide Flow-Based Services

Network layer service models:

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Guarantees?</th>
<th>Bandwidth</th>
<th>Loss</th>
<th>Order</th>
<th>Timing</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no (inferred via loss)</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
Virtual circuits: signaling protocols

- used to setup, maintain, teardown VC
- used in ATM, frame-relay, X.25
- not used in today’s Internet

1. Initiate call
2. incoming call
3. Accept call
4. Call connected
5. Data flow begins
6. Receive data

Virtual circuits

- **Signaling**: call setup, teardown for each call *before* data can flow
- **Addressing**: each packet carries VC identifier (not destination host address)
- **Router State**: every router on source-dest path maintains “state” for each passing connection
- **Resource Allocation**: link, router resources (bandwidth, buffers) may be *allocated* to VC
VC implementation

A VC consists of:
1. Path from source to destination
2. VC numbers, one number for each link along path
3. Entries in forwarding tables in routers along path
   - Packet belonging to VC carries a VC number.
   - VC number must be changed on each link.
     - New VC number comes from forwarding table
     - This is label switching!

<table>
<thead>
<tr>
<th>Forwarding table in northwest router:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Incoming interface</strong></td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Routers maintain connection state information
Allocating Resources

- Circuit switched networks allocate resources by allocating timeslots during which no other flow can use the link.
- Virtual Circuit networks use:
  - Scheduling
  - Policing
  - Drop Policies
  - Call Admission
- What are these things?
  - Let’s look at a simple motivating example

Example

- 0.5Mbps IP phone, FTP share 1.5 Mbps link.
  - bursts of FTP can congest router, cause audio loss
  - This is because the link at R1 is using First Come First Serve

Classification and Scheduling

packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly
Example

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:

Example

- Allocating fixed (non-sharable) bandwidth to flow: inefficient use of bandwidth if flows doesn’t use its allocation
  - while providing isolation, it is desirable to use resources as efficiently as possible
**Example**

- *Basic fact of life:* cannot support traffic demands beyond link capacity

**Call Admission**

Flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

---

**Mechanisms to Improve Best Effort**

- Classification and Scheduling
- Drop Policies
- Policing
- Call admission
- Implementing even a subset of these can help!

Next Question….

- How should we use these mechanisms to improve performance?
What is the “fair” allocation: (0.55Mb/s, 0.55Mb/s) or (0.1Mb/s, 1Mb/s)?

Now, what is the “fair” allocation?
Fairness

Max-Min Fair Allocation

Want to treat all the flows as equally as possible.

Give C the full 0.2Mb/s
A and B get 0.45Mb/s each
(0.45, 0.45, 0.2)

Another Example

Allocate 0.1 to C. This leaves 0.9 for the rest.
Each of A, B and D are > 0.3 so allocate them each 0.3
Max-Min Fair Allocation is (.3,.3,.1,.3)
Max-Min Fairness Algorithm

\( N \) flows share a link of rate \( C \). Flow \( f \) wishes to send at rate \( W(f) \), and is allocated rate \( R(f) \).

1. Pick the flow, \( f \), with the smallest desired rate.
2. If \( W(f) < C/N \), then set \( R(f) = W(f) \).
3. If \( W(f) > C/N \), then set \( R(f) = C/N \).
4. Set \( N = N - 1 \). \( C = C - R(f) \).
5. If \( N > 0 \) goto 1.

Example Revisted

\[ \begin{array}{c}
\text{Round 1: Set } R(f_C) = 0.1 \\
\text{Round 2: Set } R(f_B) = 0.9/3 = 0.3 \\
\text{Round 3: Set } R(f_D) = 0.6/2 = 0.3 \\
\text{Round 4: Set } R(f_A) = 0.3/1 = 0.3
\end{array} \]
Scheduling Goals

- **Flexibility**
  - Must be able to accommodate a wide range of performance objectives
  - Must be “fair”

- **Predictable/Analyzeable**
  - Must have some way to determine if the performance objectives are met

- **Implementable**
  - Cost
  - Performance
Priority Scheduling

Transmit highest priority queued packet

- multiple classes, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..
  - Problem: Higher priorities can hog

Round Robin

- multiple classes
- cyclically scan class queues, serving one from each class (if available)
- Problems:
  - What if the flows require different rates?
  - What if the packet sizes are not equal?
Fair Queueing

- This treats all the flows equally but allows for unequal packet sizes
- Flows are scheduled one bit at a time, in a round-robin fashion.
- This is called Bit-by-Bit Fair Queueing or Processor Sharing

[Diagram of Fair Queueing]

Weighted Bit-by-Bit Fair Queueing

Likewise, flows can be allocated different rates by servicing a different number of bits for each flow during each round.

\[
\begin{align*}
R(f_1) &= 0.1 \\
R(f_2) &= 0.3 \\
R(f_3) &= 0.3 \\
R(f_4) &= 0.3 \\
R(f_5) &= 0.7
\end{align*}
\]

Order of service for the four queues: ...f_1, f_2, f_2, f_3, f_3, f_3, f_4, f_4, f_4, f_1, ...

Also called "Generalized Processor Sharing (GPS)"
Packetized Weighted Fair Queueing (WFQ)

Problem: We need to serve a whole packet at a time.

Solution:
1. Determine what time a packet, \( p \), would complete if we served flows bit-by-bit. Call this the packet’s finishing time, \( F_p \).
2. Serve packets in the order of increasing finishing time.

Theorem: Packet \( p \) will depart before \( F_p + (\text{max packet transmission delay}) \)

Also called "Packetized Generalized Processor Sharing (PGPS)"

Generalized Processor sharing can be used to control the delay at a router

- Example: Two Flows share a link of 1Mbs
  - Both flows have large chunks to send
  - Case 1: Treat them equally

- Class 1: \( w_1 = w_2 \)
  - Delay = 4

- Class 2
  - Delay = 5
Adjust weights to change delay

- Suppose we want to reduce flow 2’s delay to 4

\[ 3w_1 = w_2 \]

Let’s look at what happens when serve packets not bits

\[ w_1 = w_2 \]

In this example, no packets finish later under WFQ
WFQ Example II

- Packets out of order
  - Packet from Flow 1 finishes late under WFQ
  - Theorem: Maximum lateness is \( L_{\text{max}}/C \) where \( L_{\text{max}} \) is the maximum packet size allowed. WFQ never falls further behind.

The use of WFQ for (weighted) fairness

- WFQ can be used to provide different rates to different flows.
- Most routers today implement WFQ and can be used to give different rates to different flows. (Not used much yet).
- Different definitions of a flow are possible: Application flow, all packets to a destination, all packets from a source, all http packets, the CEO’s traffic, … etc.
Policing Mechanisms

Goal: limit traffic to not exceed declared parameters

Three common-used criteria:

- (Long term) Average Rate: how many pkts can be sent per unit time (in the long run)
  - crucial question: what is the interval length: 100 packets per sec or 6000 packets per min have same average!
- Peak Rate: e.g., 6000 pkts per min. (ppm) avg.; 1500 ppm peak rate
- (Max.) Burst Size: max. number of pkts sent consecutively (with no intervening idle)

Token Bucket: limit input to specified Burst Size and Average Rate.

- bucket can hold b tokens
- tokens generated at rate $r$ tokens/sec unless bucket full
- over interval of length $t$: number of packets admitted less than or equal to $(r t + b)$. 
Policing Mechanisms (more)

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., QoS guarantee!

\[ D_{\text{max}} = \frac{b}{R} \]

Summary

- Best Effort can be improved significantly through the addition of network layer flows
- Virtual circuits implement flows
- Even in the absence of flows, router mechanisms such as scheduling and intelligent drop policies can improve performance significantly
- Next time: Quality of Service in the internet