Chapter VI: CONTROLLING NETWORKS

1. Introduction [1, 2]

Need for flow control

1.1 Rate matching

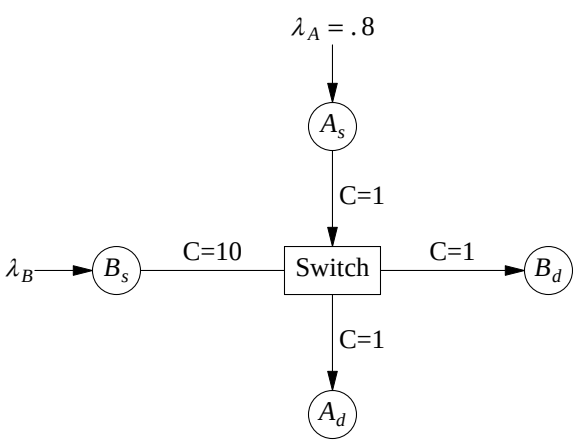
When the source can transmit data faster than the receiver can accept data, the source must be slowed down or the data will build up in the network, use up the available buffers, and prevent other sources from transmitting

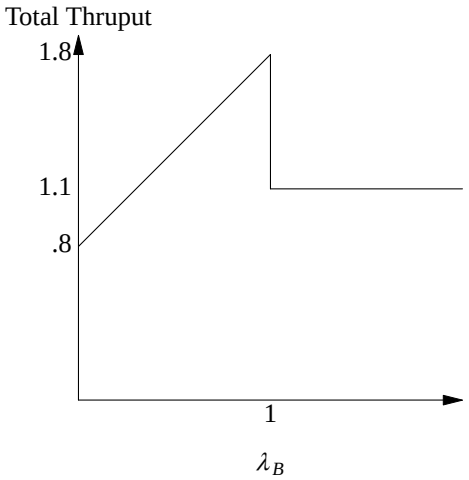
1.2 Buffer exhaustion

When the offered load is greater than can be handled, even with optimal routing, queue sizes at the bottleneck links grow indefinitely.
When the buffer is full, packets must be retransmitted, wasting network resources, and possibly reducing the throughput.

1.2.1 Simple example

finite buffer in switch





- Sources A,B retransmit messages that are lost as quickly as possible
- For $\lambda_B < 1$, everything that arrives at the switch is forwarded
- for $\lambda_B > 1$, arrivals from B cannot be forwarded as fast as they arrive, and the buffer in the switch becomes full
- Whenever a buffer in the switch becomes available, the next arrival gets the buffer.
- buffers become available at the rate $F_{A,out} + F_{B,out}$
- $F_{A,out} = \min(F_{A,in}, 1)$ and $F_{B,out} = \min(F_{B,in}, 1)$
- When the buffer is full, packets from both sources are dropped, and the sources retransmit those packets as quickly as possible.
- The input rate at the switch from B is ten times the input rate from A therefore B gets the buffer 10 times as often as A, and $F_{B,in} = 10F_{A,in}$
- Result: $F_{B,in} = 1$, $F_{A,in} = .1$, and $F_T = 1.1$

One solution: Buffer management

reserves some buffers for each path, so that none of the paths are blocked by others

1.2.2 Basis for TCP flow control

- In the Internet with fiber-optic transmission links we assume that message losses are more likely to occur because of buffer overflow than because of transmission errors.
- Each time there is a loss, the input rate of the user is decreased, to reduce the probability of buffer overflow

1.3 To provide quality of service guarantees

1.3.1 Throughput requirement:

Admission control to keep arrival rate at the network below the throughput of any link

- If each source requires 10 packets per second, and a link carries 100 packets per second, then no more than 10 sources can share any link
- If a source requests entry to the network, and there are already 10 sources on one of the links that it must use, then the source must be denied access or all of the sources on that link will fall below their required throughputs.
- If the arrival rate at the source is not deterministic, and there are a finite number of buffers at the intermediate nodes, then the sum of the rates must be far enough below 1 to make the probability of loss acceptable.

1.3.2 Delay requirement:

Priority queueing - when there are sources with different delay requirements

1. primitive:
    - sources that can sustain higher delays are given lower priority
2. more sophisticated:
    - Track the time that the packet must arrive at the destination.
    - Packets that have a longer time until they must reach the destination are given a lower priority
    - Packet discard: If packets must be discarded, discard those that have less of a chance to reach the destination in time

2. Fairness [3, 4, 5, 6]

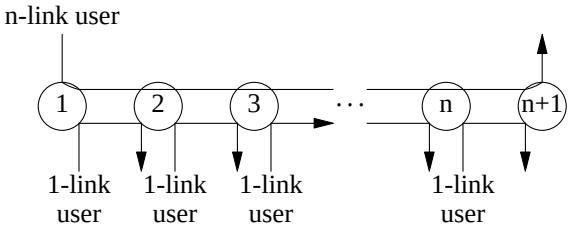
Objective: Cut back traffic fairly

Problems:

1. Different requirements:
   - Different priorities
   - Different service guarantees:
     - — minimum guaranteed rate and upper bound on delay - packet voice
     - — maximum discard rate - data

2. Different objectives

   a. 1 user traverses n links, n users traverse 1 link



   i. max throughput - n-link user gets 0 units, 1-link users each get 1 unit, Throughput = n units

   ii. Equal throughputs - each user gets 1/2 unit, Throughput = (n+1)/2 units

   iii. Equal use of network resources - 1-link user gets n/(n+1) units, n-link user gets 1/(n+1) units -- each user gets n/(n+1) link units, Throughput = $(n^2 + 1)/(n + 1)$ units

   b. In an 802.11 network: user A is close to its destination and transmits at 56 MBPS, and user B is far from its destination and transmits at 1 MBPS

      i. Equal use of the channel
      Each user has the channel at most 1/2 of the time.
      User A receives a throughput up to 23 MBPS, and user B receives up to 500 KBPS, for a total throughput of 23.5 MBPS

      ii. Each user has the same throughput:
      $x * 10^6 = (1 - x) * 56 * 10^6$, $x = \frac{56}{57}$
      User A has the channel slightly less than 2% of the time user B has the channel slightly more than 98%. Each receives about 980 KBPS, for a total throughput of 1.96 MBPS.

   c. What is the appropriate definition of fairness?

   d. How do we get it?

2.1 Max-Min Fairness

- User i obtains flow λ_i on every link it traverses
- The objective is to

   a. Give every user as large a flow as possible, subject to the capacity constraints of the links

   — For each link r, $\sum_{\lambda_i \in r} \lambda_i \le C_r$

   — A vector of flows $\vec{\lambda}$ is feasible if $\sum_{\lambda_i \in r} \lambda_i \le C_r$ on every link r

b. And, we cannot increase any flow without decreasing a smaller flow

— Formally, if a set of flows between users, $\vec{\lambda}$, is max-min fair, for any other feasible set of flows between those users, $\vec{\gamma}$, if flow $\gamma_s > \lambda_s$, then there must exist a another flow $\lambda_t < \lambda_s$ for which $\gamma_t < \lambda_t$.

- In the example, max-min fairness gives each user a flow of 1/2, and the network throughput is $\frac{n+1}{2}$

2.2 Utility Functions

Frank Kelly presented a unified approach for using utility functions to assign rates to flow [7]. The utility functions use concepts from game theory and conventional optimization techniques. The only constraints on utility functions is that they are continously increasing, concave and differentable. In other words, the higher the rate we obtain, the more useful the network becomes, but increasing the rate when the rate is low is more useful than when it is higher. He also assumes that utilities are additive, so that the aggregate utility for all sources is $U = \sum_s U_s(\lambda_s)$, where λ_s is the rate that source s obtains. The rates assigned to the sources are constrained to the feasible rates. Kelly shows that this is a conventional optimization problem that can be solved using Lagrangian multipliers.

Proportional fairness and TCP fairness are special cases of utility function fairness, and max-min fairness is a limiting case.

2.3 Proportional Fairness

- A feasible set of flows, $\vec{\lambda}$, is defined as being proportionally fair if for any other set of feasible flows, $\vec{\gamma}$, $\sum_s \frac{\gamma_s - \lambda_s}{\lambda_s} \leq 0$.
  - — Without the denominator in the summation, we are maximizing the total throughput.
  - — With the denominator the higher flows are given less weight and a zero flow is given infinite weight, so that we never set a flow to zero to increase the throughput.
  - — We weight the flows inversely in proportion to their values, which tends to move toward equal flows.
- The logarithm function satisfies the constraints for a utility function, and its derivative satisfies the definition of proportional fairness. Therefore, we can find a proportional fair system by maximizing $U = \sum_{\text{flows}} \log(\lambda_i)$ subject to the capacity constraints.
  - — The maximum of the utility function occurs at $\frac{dU(\vec{\lambda})}{dx}$
  - — $\frac{d\log(\lambda_i)}{d\lambda_i} = \frac{1}{\lambda_i}$, and
  - — Deviations from the maximum value of $\vec{\lambda}$ results in reductions in the utility function
- In the example,
  - — λ_1 is the flow for each of the 1-hop flows, and λ_n is the flow of the n-hop flow.
  - — The constraint on each of the links is $\lambda_1 + \lambda_n = 1$
  - — $U = \log(\lambda_n) + nlog(1 - \lambda_n)$
    $\frac{dU}{d\lambda_n} = \frac{1}{\lambda_n} - \frac{n}{1-\lambda_n} = 0$

    $\lambda_n = \frac{1}{n+1}$, and $\lambda_1 = \frac{n}{n+1}$
  - — The flow for λ_n is less than for max-min fairness, but the throughput increases to $\frac{n^2+1}{n+1}$

3. Flow Control
3.1 Max-Min Flow Control

Reference 1, section 6.5.2.

Fair Sharing of All of the links in a Network
When

— There are multiple source-destination pairs (P) in the network, and

— Each $p \in P$ follows one fixed path.

The flow on link "a" is

- $F_a = \sum_{\text{all sessions } p \text{ on link } a} r_p$
- C_a is the capacity of link a

Allocated rates are feasible if:

1. $r_p \geq 0$ for all $p \in P$, and
2. $F_a \leq C_a$ for all $a \in A$

A vector of rates r is max-min fair if

1. It is feasible and
2. For each $p \in P$ r_p cannot be increased, and remain feasible without decreasing $r_{p'}$ for a session p' for which $r_{p'} < r_p$

2) implies that

a. A flow isn't constained by a flow that obtains more bandwidth
if it is we can increase the flow r_p and decrease the flow $r_{p'}$ where $r_{p'} > r_p$, and

b. The network is carrying as much flow as it can under the constraints
If the network isn't, we can increase one or more flows without decreasing any others

Definition: Bottleneck link for session p:
A link on which

- $F_a = C_a$,
the link cannot carry more flow,
- and, $r_p \geq r_{p'}$ for all of the other sessions p' on the link.
We cannot increase our flow on the link without decreasing the flow of a user with less flow than us.

Proposition: a feasible rate vector is max-min fair iff each session has a bottleneck link

If not, we can increase the rate of the session without decreasing the rate of a session with a lower rate.

3.1.1 The Water Filling Algorithm

3.1.1.1 Approach

1. Maximize the network use allocated to the session with the minimum allocation
2. Once the use of the most constrained session is maximized, maximize the allocation of the next most constained session.

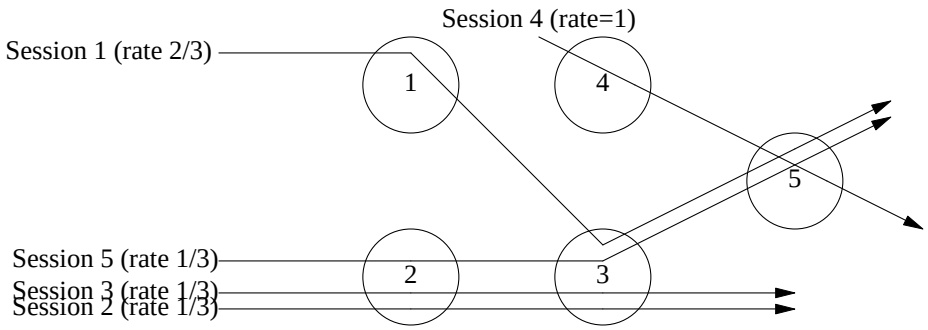
3. ⋯

3.1.1.2 Algorithm

1. All flow are initially 0
2. Increment all flows until one or more links become bottlenecked.
3. Fix the flows on the bottlenecked links.
4. Increment all flows that aren't fixed until one or more additional links become bottlenecked.
5. Fix the flows on the new bottlenecked links.
6. Proceed until all flows have a bottleneck link.

3.1.1.3 Example:

All of the link capacities are 1



1. Increase all flows to 1/3
   - Link 2-3 becomes saturated and is the bottleneck link for sessions 2,3,5
   - Fix the flows of these sessions to 1/3
2. Increase flows for sessions 1,4 to 2/3
   - Link 3-5 becomes saturated and is the bottleneck link for session 1
   - Fix the flow os session 1 to 2/3
3. Increase the flow for session 4 to 1
   - Link 4-5 becomes saturated
   - Fix the flow for session 4 to 1

3.1.1.4 Modifications to the algorithm

1. *Bounded Link Delays*
   Set the capacity of a link to $C_a' < C_a$ to bound the link delay.
   When the flow on a link approaches the capacity of a link, the queuing delay can become infinite.
2. *Priorities*:
   Assign priorities to some sources to provide them with greater flows than others.
   Increase the high priority flows faster than others
3. *Assign max rates to each of the flows*
   The initial implementation assumes that sources can always use more capacity, *The flows are unconstrained*.
   In reality if flows reach the limit that the user presents to the network, then the user isn't constrained by the network.
   When a flow reaches its upper limit, stop increasing that flow

Constrained flows can be modelled by inserting a channel between each constrained source and the network, where the channel capacity is the user's maximum rate.

4. *Max-Min fairness on a multicast tree:* [5]
   The flow is transmitted on a multicast tree, rather then a point-to-point path.
   The objective is to give the same rate to all of the multicast receivers.
   Increase the rate on all of the links on the multicast tree until any link saturates, then stop.

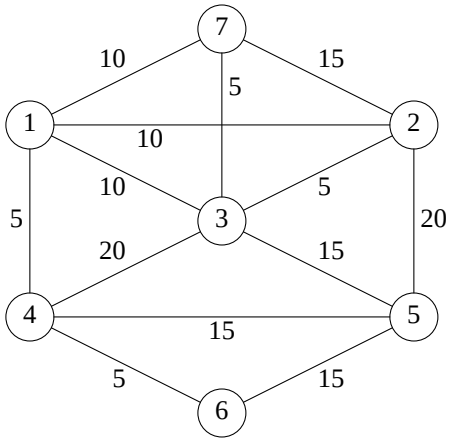
5. *Max-Min fairness with proportional routing:*
   A fixed proportion of the flow from a source is transmitted on different paths. The source uses several paths in the network
   Increase the total flow for a source at the rate of the other sources, but increase the flow on each of the paths by the fraction of the flow on that path.
   Stop increasing the flow for the source when any of the paths saturates.
   The paths assigned to a source may have some links in common.

   This is similar to priorities.
   Each sub-flow is assgned a priority equal to the fraction of the flow on the link. The flows with lower priorities are increased more slowly.
   The difference between simple priorities and proportional routing is that the sub-flows are linked together. When one stops incresing, they all stop increasing.

   A more general problem does not set the proportions, but allows us to change the proportions.
   We can lower the proportion of a source's flow on a constrained path in order to increase the flow obtained by the source.
   The max-min fair assignment of proportions cannot be solved using the water filling algorithm, but there is a linear programming solution

Home work

Max-min fair flow control
The network has bidirectional links with the capacity shown in each direction.



The following flows are placed on the paths shown. r_{max} is the maximum rate that the flow can supply.

Flow	Path	r_{max}
F_A	6->5->3->1	10
F_B	2->5->3->4	7
F_C	2->3->4	10
F_D	1->3->4	10

Use Max-min fair flow control to determine the rate allowed for each of the flows

3.1.2 Bottleneck flow control [8]

- Fairness objective: To guarantee that a user gets as much flow through his bottleneck links as any other user on those links.
- The objective is similar to Water Filling Algorithm. However:

  A. Water Filling

     1. Is a centrallized algorithm that allocates bandwidth to each user
     2. If the users change, the algorithm is rerun
     3. The entire bandwidth can be allocated, although less bandwidth may also be allocated to control the delay in the network

  B. Bottleneck flow control

     1. is distributed - each source adjusts its own rate based upon measurements that it makes on its own path
     2. the algorithm runs continuously, - so that the share of the bandwidth that a user obtains changes as users enter and leave the network
     3. the algorithm is asynchronous - the sources don't coordinate the time that they make changes
     4. The algorithm uses the unused bandwidth on each link to control the bandwidth that each source obtains, and cannot assign all of the bandwidth on the links.

- The technique is similar to bandwidth balancing in DQDB.
  - The systems do not allocate all of the bandwidth on a channel.
    A user is constrained to a fraction, less than 1, of the bandwidth that is not being used by the other users on a channel.
  - Bottleneck flow control differs from BWB because a flow passes over several channels.
    A user is limited by the flows on its most constrained channel.

Terms

- C_k = capacity of link k
- γ_r = the flow obtained by the r^{th} user.
- $f_k = \sum_{r \in k} \gamma_r$ = total flow on k^{th} link
- $R_k = C_k - f_k$ = unused, residual, capacity on link k
- $\alpha_{r,k}$ = the fraction of the bandwidth that is not used by other users on a channel k that user r can acquire.
  - $0 < \alpha_{r,k} < 1$
  - By giving some users a larger value of α, those users receive a larger fraction of the available bandwidth, and have a higher priority.
    (ie. some user may take 90% of the unused bandwidth, and other users may only take 1/2
  - Different values of α are used on different links to control the delay on the links and thereby control the network delay.
    The link delays depend on both the α's of the users on the link and the number of users. We can control the delay by using smaller values of α on links with a larger number of active users.

- $\gamma_{r,k} = \alpha_r(R_k + \gamma_r)$ is the maximum bandwidth that user r can acquire on link k.
  - γ_r is the value user r's bandwidth that is used to calculate f_k and therefore R_k.
  - Reference [8] uses a value $0 < X_{r,k} < \infty$ instead of $\alpha_{r,k}$, with $\gamma_{r,k} = X_{r,k} R_k$. $\alpha_k = \frac{X_k}{1 + X_k}$. The calculations are equivalent, but we use α because it is more intuitive and is closer to bandwidth balancing.
- γ_r^0 = the maximum flow that the r^{th} user needs, or the maximum flow that it is allowed to acquire by a contract.
  - a voice source may set γ_r^0 to the bit rate that it would like to use.
  - A data source may set $\gamma_r^0 = \infty$ to get as many bits/sec as possible.
  - When $\gamma_r^0 = \infty$ the user is an "unconstrained user"

Technique:

User r learns $\gamma_{r,k}$ for each link on its path, and adjusts its flow to $\gamma_r = \min\left(\gamma_r^0, \min_{k \in p}\left(\gamma_{r,k}\right)\right)$

Definition of Bottleneck Link:
When $\gamma_r = \gamma_{r,k}$, user r is constrained by link k, and link k is user r's bottleneck link.

Fairness::
When $\alpha_{r,k} = \alpha_k$ for all r, k, there are no high priority users.

- The users that are constrained by link k, eventually all obtain the same flow, $\gamma_r = \alpha_k(R_k + \gamma_r) = \frac{\alpha_k}{1 - \alpha_k} R_k$.
  The flows on the link are adjusted asynchronously, but converge to the same value the same way as in bandwidth balancing.
- and, all of the users that share the link, but are not constrained by it, receive a smaller flow, guaranteed by the minimization.

3.1.2.1 Relationship between $\alpha_{r,k}$ and delay

Consider a link k with N_k users, all of whom have the same priority so that $\alpha_{r,k} = \alpha_k$

- The normalized delay, using Kleinrock's independence approximation, is:

$$\bar{D}_{k,\text{norm}} = \frac{\bar{D}_k}{1/(\mu C)} = \frac{1}{(1-\rho_k)} = \frac{C_k}{C_k - f_k} = \frac{C_k}{R_k}$$

- The max. flow occurs when all N_k users are all bottlenecked on this link

$$\gamma_r = = \frac{\alpha_k}{1-\alpha_k} R_k$$

$$f_{k,max} = N_k \gamma_r = \frac{N_k \alpha_k}{1-\alpha_k} (C_k - f_{k,max})$$

$$= \frac{N_k \alpha_k}{1 + (N_k - 1)\alpha_k} C_k$$

- The largest assigned flow yields the smallest residual capacity:

$$R_{k,\min} = C_k - f_{k,\max} = \frac{1-\alpha_k}{1 + (N_k - 1)\alpha_k} C_k$$

- And, the smallest residual capacity yields the largest queueing delay:

$$\bar{D}_{k,\text{norm,max}} = \frac{1 + (N_k - 1)\alpha_k}{1-\alpha_k}$$

- When $\alpha_k = .9$, and 5 users are sharing the link, $D_{k,norm} = 46$ message transmission times.
- In order to guarantee that the maximum delay on a link is $\leq \bar{D}_{\max}$,

$$\alpha_k \leq \frac{\bar{D}_{\max} - 1}{N_k + (\bar{D}_{\max} - 1)}$$

The fraction of the unused capacity that a source requests must decrease as the number of users sharing the link increases.

Gerla and Staskauskas, reference 9 extended bottleneck flow control to cut-set flow control to limit the flow across a cut instead of a single path.
A saturated user traverses a saturated "cut" on which his flow ≥ the flow of any other user on that cut

Conzhou Zhou and Maxemchuk, reference 10 extended bottelneck flow control to MANET's
The residual capacity at a node is the fraction of the bandwidth not being used by an of the nodes in a transmission region.

3.1.2.2 Asynchronous Distributed Implementation:

- Each packet contains:

  — a field in which the source, r, inserts γ_r, its current flow rate, and

  — a field, $\gamma_{r,\min}$, that each node that the packet passes through can write.
  When the packet enters the network, $\gamma_{r,\min} = \gamma_r^0$, the maximum rate that the source can supply.

- At each node, k,:

  — The node calculates:

  1. $R_r = C_k - \sum_{i \neq r} \gamma_{i,store}$, the residual capacity available to source r.
     - The $\gamma_{i,store}$ are the flows of the sources that traverse this node.
     - If a node is inactive for a specified amount of time, its flow is set to zero.
  2. $\gamma_{r,k} = \alpha_{r,k} R_r$, is the maximum capacity that source r can acquire at node k.

  — The node lowers the value $\gamma_{r,min}$ stored in the packet to $\min\left(\gamma_{r,min}, \gamma_{r,k}\right)$

  — The node changes its stored value of $\gamma_{r,stored} = \min\left(\gamma_r, \gamma_{r,min}\right)$.

- At the packet destination the value of $\gamma_{r,\min}$ is returned to source, r, in the acknowledgement, and the source adjusts its rate to $\gamma_r = \gamma_{r,\min}$.

3.1.2.3 Synchronous Calculation of the Flows

- The distributed implementation converges toward a solution.
- When all of the flows on a link have the same α's, the final solution assigns the same capacity to each flow that is constrained by this link, and all of the other flows on this link, that are more severley constrained by another link, get a smaller capacity.
- The technique for determining the final capacity is similar to the technique used for bandwidth balancing:
  The capacity that is left on the link for the n flows that are constrained by this link, after the flows that are more constrained by another link are subtracted is
  $C'_k = C_k - \sum_{r \in others} \gamma_r$.

  — If all of the flows on the link use the same α_k, the flows converge to:
  $$\gamma_{r,k} = \alpha_k\left(C'_k - (n-1)\gamma_{r,k}\right) = \frac{\alpha_k C'_k}{1 + (n-1)\alpha_k}$$

  — If the flows that are constrained by the link use different α_k's, to obtain priorities, then we must solve a set of simultaneous equations to determine the values to which the distributed solution converges.
  For instance, if n_1 of the bottlenecked flows have α_{k1}, and n_2 have α_{k2}, the simultaneous equations that we must solve to determine the convergent values $\gamma_{r1,k}$ and $\gamma_{r2,k}$ are:

  $$\gamma_{r1,k} = \alpha_{k1}\left(C'_k - (n_1 - 1)\gamma_{r1,k} - n_2\gamma_{r2,k}\right)$$
  $$\gamma_{r2,k} = \alpha_{k2}\left(C'_k - n_1\gamma_{r1,k} - (n_2 - 1)\gamma_{r2,k}\right)$$

- Solving for the convergent values is complicated by the fact that, initially we don’t know which flows are constrained by which links.
  We resolve this problem with an iterative algorithm:

  — The algorithm is similar to water filling. In each step we find the next flow or set of flows that reach their maximum value, and cap the flows at that value.
  The flows that have reached their maximum value are called "saturated" and the flows that have not reached their maximum value are called "unsaturated."

  — The algorithm is different than the algorithm described for water filling in that we do not increase the flows incrementally, but instead calculate the next value of flows that may cause flows on a particular link to saturate.

  — Initially,
  All of the flows, γ_r, are unsaturated.
  The capacity available for unsaturated links k is the capacity of the link, C_k.

  — As flows determine their maximum fair value and are saturated, $\gamma_r = \gamma_{r,sat}$, the capacity available for the remaining unsaturated flows on the link is reduced to $C'_k = C_k - \sum_{r \in k} \gamma_{r,sat}$.

  — On each link k we calculate the flow $\gamma_{r,k}$, as determined by the number of unconstrained flows, their $\alpha_{r,k}$, and the capacity available for the unconstrained flows,C'_k.

  — A source obtains the smallest flow on its path, or the maximum flow, γ_r^0, that is delivered by the source.
  $\gamma_r = \min\left[\gamma_r^0, \min_k \gamma_{r,k}\right]$

  — When some of the flows on a link are more severely constrained by other links, or are more severely constrained by the maximum flow that they can supply, the remaining flows share the bandwidth that they could not use, and receive additional bandwidth.

  — If all of the flows on a link are constrained by the link, or if a flow is limited by the amount that a source can supply, these flows have reached their maximum capacity, and are saturated. We can fix the solution for these flows and subtract them from the capacity on all of the links that they traverse.

  — We repeat the calculations with the C'_k adjusted for the saturated flows and find the links where all of the remaining flows on the link are constrained by that link, or the flows are constrained by the source. These flows have reached their saturated capacity, and can no longer increase. We remove these flow from the link capacities, and repeat the procedure until all of the flows are saturated.
  Reference [8] has shown that at least one flow will become saturated at each iteration, so that the algorithm will terminate.

Algorithm: For $\alpha_{r,k} = \alpha_k$:

- Initially,
  - For each user, r:
    - Their final flow is not known.
    - Their path, the links that they use, is known.
  - For each link, k:
    - The number of users that may be constrained by link k is $n_k(0)$, the number of users whose flow is routed over the link.
    - The flow of the saturated users on the link is $F_{k,sat}(0) = 0$.
- On the i^{th} iteration,
  - For each link, k, with at least one unsaturated user,
    - Calculate the flows for the saturated users on the link, $F_{k,sat}(i) = \sum_{r \in k} \gamma_{r,sat}$
    - Count the number of unsaturated flows on the link, $n_k(i)$.
    - Calculate the flow available for each unsaturated user as:

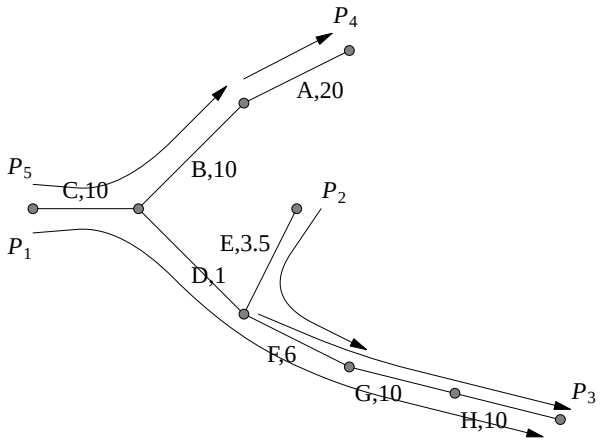
$$\gamma_{r,k}(i) = \alpha_k\left(C_k - F_{k,sat}(i) - (n_k(i) - 1)\gamma_{r,k}(i)\right)$$

$$= \alpha_k\left(\frac{C_k - F_{k,sat}(i)}{1 + (n_k(i) - 1)\alpha_k}\right)$$

  - For each user, r:
    - Calculate $\gamma_r(i) = \min\left(\gamma_r^0, \min_{k \text{ on } path} \gamma_k(i)\right)$,
  - For each link, k:
    - If $\gamma_r(i) = \gamma_{r,k}(i)$ For all of the $n_k(i)$ unsaturated users, $\gamma_{r,sat} = \gamma_r(i)$, and the users are saturated.
  - For each unsaturated user, r,
    - If $\gamma_r(i) = \gamma_r^0$, a user is constrained by his requirement and $\gamma_{r,sat} = \gamma_r^0$
- This algorithm will terminate
  - The capacty offered to each unconstrained user can only increase on each iteration.
  - In each iteration at least one user will saturate

3.1.2.4 Example

$\alpha_k = .5$, the total flow from the users is unlimited, $\gamma_r^0 = \infty$ for all r.

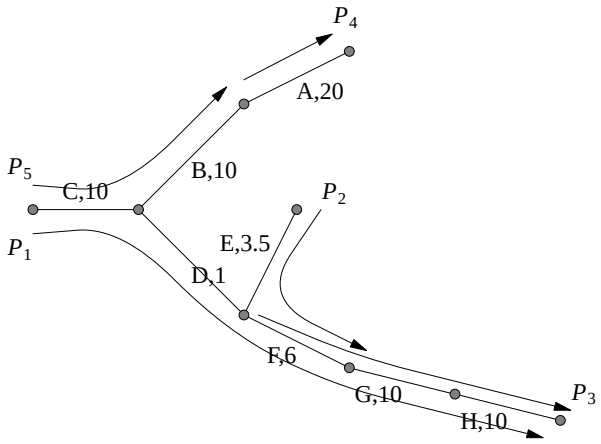


				Allowed Flows					
	$n_k(i)$	$F_{l,sat}(i)$	$\gamma_{r,k}(i)$	P_1	P_2	P_3	P_4	P_5	$\gamma_{l,sat}$
Iteration 1									
Link A	1	0	10				10		10
Link B	1	0	5					5	
Link C	2	0	10/3	10/3				10/3	
Link D	1	0	1/2	1/2					1/2
Link E	1	0	7/4		7/4				
Link F	3	0	3/2	3/2	3/2	3/2			
Link G	2	0	10/3	10/3		10/3			
Link H	2	0	10/3	10/3		10/3			
$\gamma_r(1)$				1/2	3/2	3/2	10	10/3	
$\gamma_{r,sat}$				1/2			10		
Iteration 2									
Link A	0	0	10				10		10
Link B	1	0	5					5	
Link C	1	1/2	19/4	1/2				19/4	19/4
Link D	0	0	1/2	1/2					1/2
Link E	1	0	7/4		7/4				7/4
Link F	2	1/2	11/6	1/2	11/6	11/6			
Link G	1	1/2	19/4	1/2		19/4			
Link H	1	1/2	19/4	1/2		19/4			
$\gamma_r(2)$				1/2	7/4	11/6	10	19/4	
$\gamma_{r,sat}$				1/2	7/4		10	19/4	
Iteration 3									
Link A	0	0	10				10		10
Link B	0	0	21/4					19/4	
Link C	0	1/2	19/4	1/2				19/4	19/4
Link D	0	0	1/2	1/2					1/2
Link E	0	0	7/4		7/4				7/4
Link F	1	9/4	15/8	1/2	7/4	15/8			15/8
Link G	1	1/2	19/4	1/2		19/4			
Link H	1	1/2	19/4	1/2		19/4			
$\gamma_r(3)$				1/2	7/4	15/8	10	19/4	
$\gamma_{r,sat}$				1/2	7/4	15/8	10	19/4	

- The residual capacity is the capacity that is unasigned when all of the flows on a link are saturated.
  - What is the residual capacity on links F, G, and H in the preceding example?
- With $\alpha_r = .5$, the throughputs are much less than those in in max-min flow control with link utilizations of 1.
  Similar results would be obtained if we limited the flows to a fraction of the link capacity in the max-min algorithm

Home work

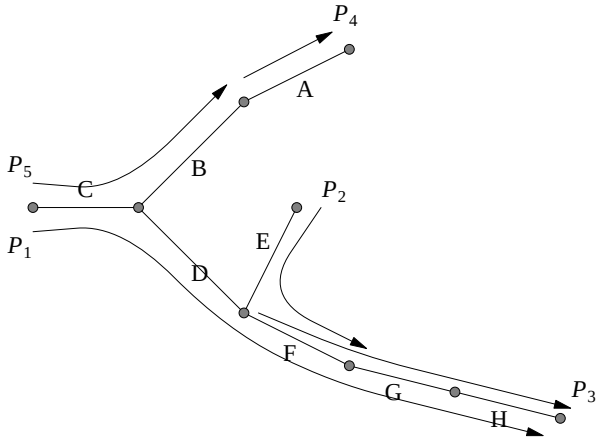
1.



Given: $\alpha_{1,k} = \alpha_{3,k} = .5$, $\alpha_{2,k} = \alpha_{4,k} = \alpha_{5,k} = .9$, $\gamma_r^0 = \infty$ for all r.

Find $\gamma_{r,k}$

2.



$\gamma_r^0 = \infty$ for all r. The capacity of each link is C bps, the average message length is 1/μ.

A. For each link, k pick the α_k so that the maximum nirmalized average delay is 3.

B. find γ_r

C. Calculate the normalized average delay on each link

3.3 Combined Optimal Proportional Routing and Flow Control

Reference [1], section 6.5.1

Model:

- There are several source-destination (S-D) pairs in a network
- Each S-D pair has a small number (possibly 1) route that it can use, and must decide how to split its flow over the designated routes

We will jointly consider the proportional routing and flow control problem

- The flows from the S-D pairs may be greater than the network is able to transfer, and we must decide both how to split the flows on the allowed paths and also how much to limit the flows from the different S-D pairs.
- This problem is more difficult than that solved in max-min fair flow control and bottleneck flow control, because we can consider moving flows between the different routes for an S-D pair.
  There isn't a single link that bottlenecks an S-D, and
  There aren't necessarily other S-D pairs that share all of the congested links with this S-D pair
- We will show that the optimal routing and flow control problem is actually the same as the optimal routing problem.

Adjust both the routing and the flows between all S-D pairs.

- S-D pair w has a required flow r_w
- It transfers r'_w over the paths $p_w \in P_w$, and
- Blocks $y_w = r_w - r'_w$ from entering the network

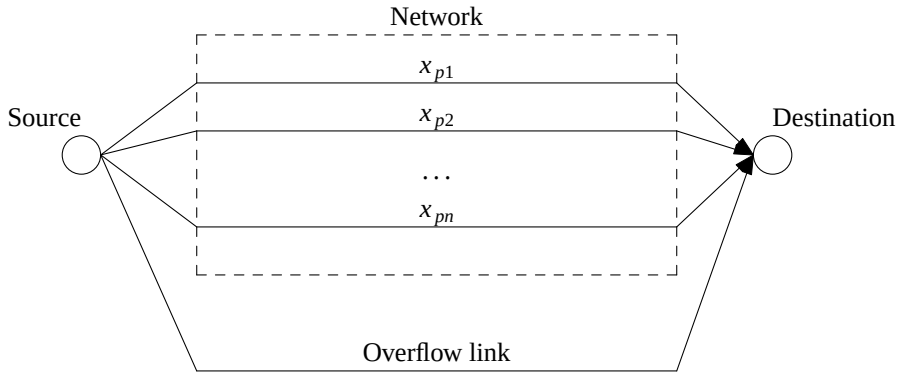
If we minimize the cost function for routing $C_N = \sum_{i,j} D_{i,j}(F_{i,j})$

and let the flows r'_w vary, so that $0 \le r'_w \le r_w$,
the minimum cost (delay) solution always occurs when $r'_w = 0$.
Nothing is transmitted on the network
Therefore, we must include a penalty function $E_w(y_w)$ for reducing r_w.

The optimization problem becomes:

- minimize $C_N = \sum_{i,j} D_{i,j}(F_{i,j}) + \sum_{w \in W} E_w(y_w)$
- subject to the constraints that:
  - $\sum_{p_w \in P_w} x_{p_w} + y_w = r_w$, for all $w \in W$
  - $x_{p_w} \ge 0$ for all $p_w \in P_w$, $w \in W$
  - $y_w \ge 0$, for all $w \in W$

This is the optimum routing problem with an extra path placed on the network that carries the traffic that is removed on each S-D w



$$y_w = r_w - r'_w$$

The combined routing and flow control problem is mathematically equivalent to the optimum routing problem

The algorithms that have been applied to optimum routing can be applied directly to the optimum routing and flow control problem

The source, w, collects data on the sum on the first derivatives on the paths p_w, calculates the first derivative $\frac{\partial E_w(y_w)}{\partial y_w}$, and moves flow to the MFDL path.

The form of the penalty function E_w

1. Determines how willing we are to remove flows rather than sustain delays on the paths
2. Determines whether or not we remove flows fairly, and
3. Can give priorities to different flows

We need to specify $\frac{\partial E_w(y_w)}{\partial y_w}$ rather than $E_w(y_w)$

A penalty function that has been suggested is

$$\frac{\partial E_w(y_w)}{\partial y_w} = \left(\frac{a_w}{r_w - y_w}\right)^{b_w}$$

This penalty function

- Has a form similar to the first derivative of the delay function with the link capacity replaced by the requested flow.
- If we try to take all of the flow away from a source, the penalty $\rightarrow \infty$
- When $b_w = 2$ delay and discarding are equally costly, but if we make $b_w > 2$, discarding becomes more costly
- If we make the a_w different for different w, we can give priorities to sources by make it more costly for some sources to discard part of their flows.

4. Fair Queueing

Reference 11, sections 7.7.2-7.7.3

Fair sharing of a single link by multiple sources.

4.1 Simple Fair Queueing

Each active source gets an equal share of the bandwidth on a link, up to its throughput

— If a source requires less than others, it gets its full requirement before the other sources get more than its full requirement.

Implementation
Each flow gets its own queue

A. With fixed size packets, like ATM, perform round robin service on the non-empty queues

B. With variable size packets

   a. Approximate bit by bit round robin.

   b. When each packet arrives calculate the time it would complete transmission if each active queue is served bit by bit round robin

   c. Each time a packet is selected for transmission, select the packet that would have completed first in a bit by bit round robin service

   d. Approximation of completion time for packet n+1, in queue Q, that arrives at time t_A is:
   $$t_Q(n+1) = \frac{L_{n+1}}{C/N_A} + \max(t_Q(n),\, t_A)$$

      - N_A is the number of queues that are active when the packet arrives
      - In bit-by-bit round robin, the source would send C/N_A bits/sec.
      - L_{n+1} is the number of bits in the packet.
      - $t_Q(n)$ is the time that the previous packet in this Q completes service, or the arrival time if the queue is empty

   e. The calculation is a very rough approximation. The completion time can be made more accurate by

      1. Taking into account that some of the queues would become inactive while the packet is being serviced in a bit-by-bit round robin
      2. By recalculating all of the packet completion times if another queue becomes active before it would have completed a bit-by-bit round robin service.

   However, the rough approximation is preferred in applications.

4.2 Weighted Fair Queueing

1. Each flow gets a different fraction of the bandwidth
2. For fixed size packets, use round robin service, but give some queues one service, others 2 services, others 3 and so on.
3. In each service round, $\sum_{i \in A} W_i$, fixed size packets are transmitted.
4. One way to order the packet transmissions in a round is to transmit W_i packets from source i, followed by W_j packets from source j, and so on.
5. Anouther way to order the packet transmission is to divide a service round into sub-rounds.
    a. Each queue i has a weight W_i, that is the number of packets that it transmits in a round.
    b. The number of sub-rounds in a round is $\max_i W_i$.
    c. On the first sub-round all active queues with $W_i \geq 1$ are serviced.
    d. On the second sub-round all active queues with $W_i \geq 2$ are serviced.
    e. And so on up to the maximum weight, then the first sub-round is started again
    f. A round lasts $\sum_{i \in A} W_i$ packet transmission times, and a queue with weight W_i transmits W_i packets.
6. For variable size packets,
    - the completion time for weighted bit-by-bit service is calculated as $t_Q(n+1) = \frac{L_{n+1}}{C\left(W_Q/(\sum_{i \in A} W_i)\right)} + \max(t_Q(n), t_A)$
    - A member of this queue transmits W_Q bits during the $\sum_{i \in A} W_i$ bits in the in a bit-by-bit service round, and obtains $C\left(W_Q/(\sum_{i \in A} W_i)\right)$ bits/sec.
    - At the beginning of each packet transmission, the queue is selected whose lead packet has the smallest completion time.

Home work

Weighted Fair Queueing

Consider a link that services 5 queues, Q_i for $i = 1, 2, 3, 4, 5$
The weights assigned to the queues are $W_1 = W_2 = 1$, $W_3 = 3$, $W_4 = W_5 = 2$.
The capacity of the link is 1000 bps

Consider the sequence of messages M_1 to M_6 specified in the table
that arrive arrive at times t_A,
have length L bits, and
are entered into queues Q_i.

Complete the table, where
W_A is the the weight of the active queues when the message arrives
$t_Q()$ is the estimated time that the message completes transmission with bit-by-bit round robin service,
t_s is the time that the message begins transmission, and
t_e is the time that the message ends transmission

What is the order in which the messages are transmitted?

Msg	Queue	t_A secs.	L (bits)	W_A	$t_Q()$	t_s	t_e
M_1	Q_1	0	1000				
M_2	Q_2	.5	1000				
M_3	Q_3	.6	1000				
M_4	Q_2	1.1	500				
M_5	Q_4	2.1	1000				
M_6	Q_5	2.2	100				

5. Congestion Control

5.1 Open Loop Control

- Does not depend on feed back information to regulate the entry of data to the network
- Assumes that once a source is admitted, the network can handle the flow as long as the source does not exceed the rate that it promised to send

5.1.1 Admission Control

Decide whether or not to admit a source to the network

Technique:

- for fixed rate sources, the sum of the rates on each link cannot exceed the link capacity
- For variable rate sources, each source specifies its average rate and peak rate.

Effective Bandwidth

- An effective rate is assigned to the source - between the peak and average - and the effective rate on each link cannot exceed the capacity of the link
- Assigning according to the peak rate is conservative. All of the sources always get through, but the link may not carry as many connections as it could
- Assigning according to the mean is aggressive. Occassionally the sources exceed the mean for a long period of time and require very large buffers or have a high probability of losing packets.
- How aggressive we can be at a certain probability of loss depends on the variance of the arrival process, which generally is not known.
- Typically, a weighting factor is chosen as $R_{eff} = \mu + \alpha * (peak - \mu)$, where $0 \leq \alpha \leq 1$.
  - — Intuitively, increase α if the loss probability is higher than is acceptable.
  - — If we increase α, we give each user more bandwidth, and assign fewer users to a link

5.1.2 Policing

- Once a source is admitted to the network, there has to be a mechanism to make sure that the source stays within the parameters that are agreed upon.
- Rather than discarding packets when a source exceeds its "contract", the network can mark the packets as low priority, and discard them later if the network is congested

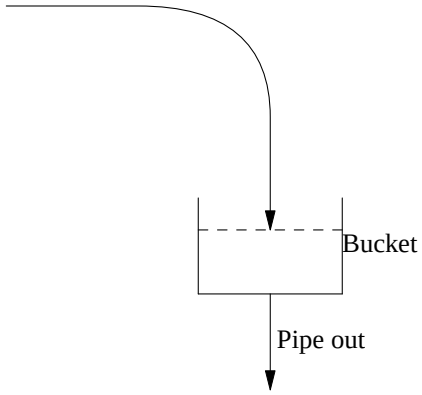
5.1.2.1 The Leaky Bucket Algorithm

Used for policing in ATM

Objective: When a user exceeds his average contracted rate, he is still allowed to transmit, but the packets that exceed the contract may be discarded in congested regions of the network

- *Over what period do we define the average?*
- *How do we implement the algorithm?*

Idea:

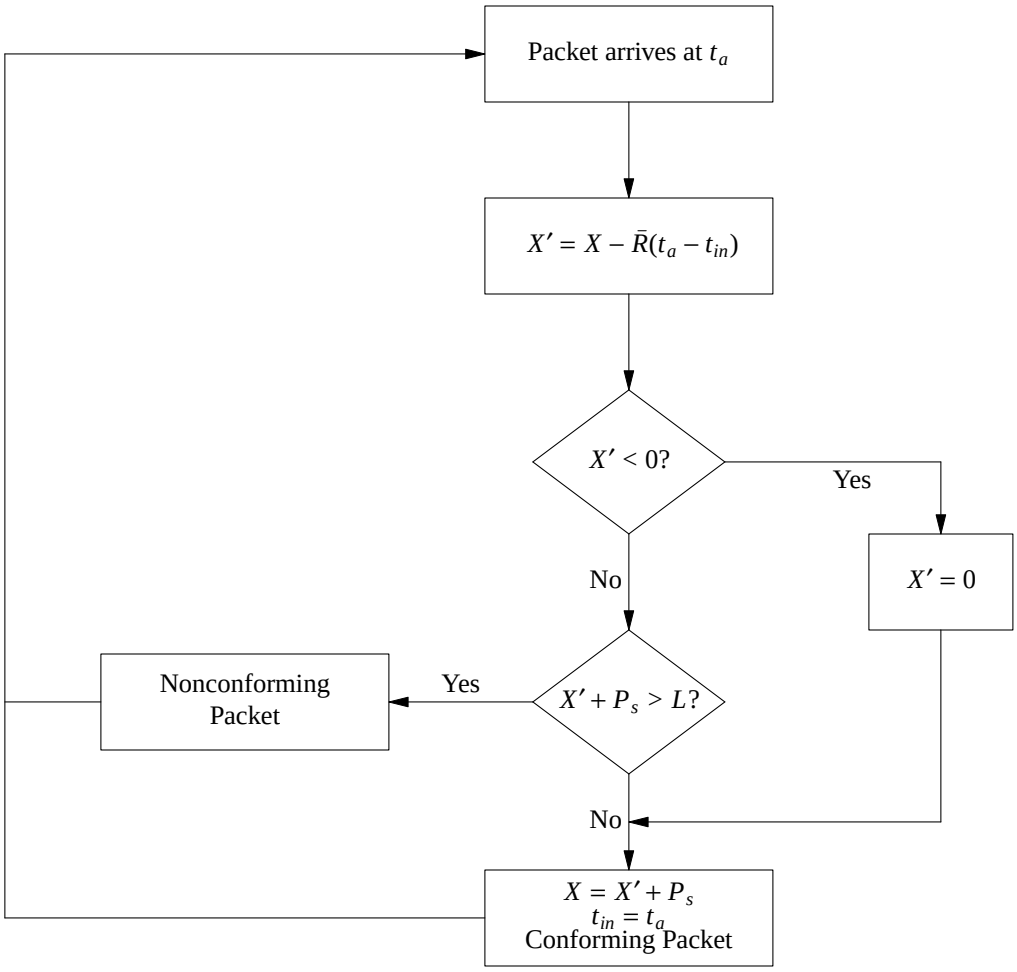


- The width of the pipe out determines the rate that the bucket can drain
  The width of the pipe is analagous to the average rate that is contracted
- The size of the bucket determines how far ahead of the output the input can get before there is an overflow
  The size of the bucket is related to the period of time that we average over

Implementation:

- All of the packets from a source are transmitted immediately.
- There is a *Conceptual* bucket that packets may be placed into.
  The bits in the bucket are drained at the source's contracted rate.
- When a packet arrives:
  - If the packet fits into the bucket it *conforms* with the contracted rate. It is unmarked.
  - If the packet doesn't fit into the bucket, it exceeds the contracted rate. It is marked as non-conforming and may be dropped in a congested region of the network.

Algorithm

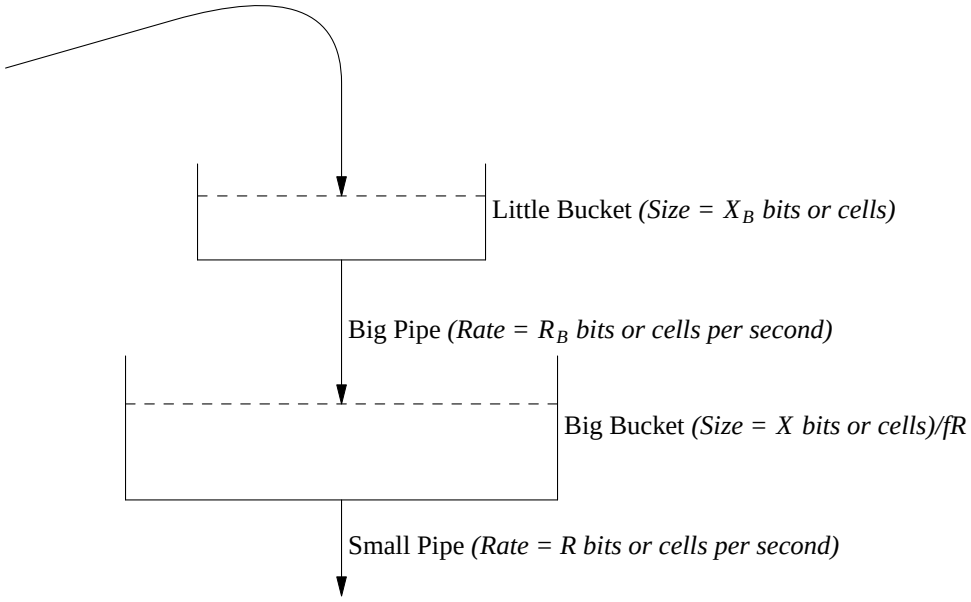


- t_{in} = the time that the last packet was placed in the bucket
  - X = the number of bits in the leaky bucket at $t_{in} + \varepsilon$
    Just after the last packet was placed in the bucket
- t_a = the time that the current packet arrives
  - X' = the number of bits in the bucket at $t_a - \varepsilon$
    Just before the current packet arrives
  - $X' = \max\left(X - \bar{R}(t_a - t_{in}),\ 0\right)$
    where $\bar{R}$ = the rate that the bucket is drained
    The average rate negotiated by the source
- If $X' + P_s > L$ the cell is not placed in the bucket, and is marked as non-conforming
  - L = the size of the bucket in bits
  - P_s = the size of the current packet
    In ATM model the packets have a fixed size. The times can be normalized to packet transmission times, the size of the bucket can be measured in cells, and the negotiated rate can be expressed as arrivals per cell transmission time.

5.1.2.2 Dual Leaky Buckets

Used to police the peak and average rates

- The leaky bucket allows the instantaneous arrival rate to exceed the contracted rate for a period of time.
  - — It can exceed the contracted rate by a large amount for a short period of time or a smaller amount for a longer period of time
  - — We can transmit a burst of packets that fill the bucket.
- Long bursts place a burden on network resources, and should be limited
- The dual leaky bucket reduces the burst size by placing the packets in a first bucket that is smaller, but has a bigger pipe.
  - — The leaky bucket may prevent the rate from exceeding the average rate by 10% for more than a second
  - — The first bucket may prevent the rate from exceeding twice the average for more than a fraction of a second.



5.1.3 Traffic Shaping

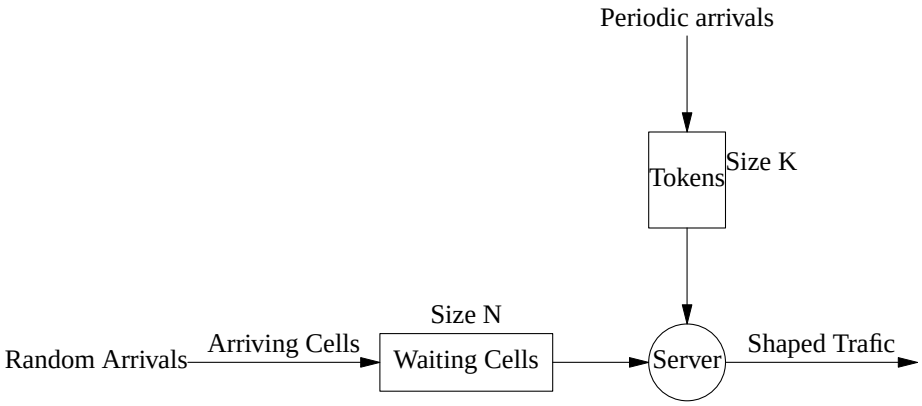
Traffic shapers smooth out bursty traffic to make it appear more periodic. This places less burden on the network to store long bursts from multiple inputs

5.1.3.1 Leaky bucket traffic shaper

- In ATM, cells from a source are placed in bucket (a local buffer). One cell is transmitted every I slots, when the buffer is not empty
  - — If the transmission rate of the line is R_L, and the negotiated rate is R, then $I = R_L/R$.
  - — $R \leq R_L$, or the line cannot support the negotiated rate.
  - — Normally, R_L/R is an integer.

- When the bucket is full, cells are discarded, or labeled for later discard
- The bucket size, determines the maximum burst size

5.1.3.2 Token bucket traffic shaper



Tokens arrive periodically and are stored up to a maximum number K

When a source is inactive it builds up transmission credits for when it becomes active

Cells arrive randomly and up to N can be stored to wait for tokens

- When there are tokens available, and a burst of cells arrive, the burstiness remains
- When there are no tokens waiting, the cells are removed from the bucket periodically as tokens arrive.

The maximum size of a arriving burst is $N + K$

The maximum size of a transmitted burst is K

When $K = 0$, the token bucket shaper becomes a leaky bucket shaper

- The two shapers trade storage and delay for smoothness, and can be set anywhere between perfectly smooth with long delays to bursty with no delay

Home work

Dual Leaky bucket policing algorithm with variable size packets

a. Draw a flow chart to implement the dual leaky bucket with variable size packets, similar to the flow chart for the leaky bucket with fixed size packets in the notes.

b. L_1 = 1000 bits, L_2 = 10,000 bits, $R_1 = 200 bits/sec.$, $R_2 = 50 bits/sec.$.
At time t=0, both buckets are empty.
Starting at t=0, fixed size packets of 100 bits arrive every .25 seconds, for an average rate of 400 bits/sec., at what time is the first packet marked as non-conforming?

c. Repeat part b when the packets arrive every second, for an average rate of 100 bits per second.

5.2 Closed Loop Control

- explicit messages, received from the network or the destination, indicate the state of the network

5.2.1 Window Flow Control

- A source can have W unacknowledged packets in the network
- When the destination receives a packet it returns a permit to send another packet
- Permits can be numbered to detect lost permits
  or a permit can be treated as an ACK in a positive acknowledgement protocol
  ie: The packet is retransmitted to recover lost permits.

5.2.1.1 End-to-end windows

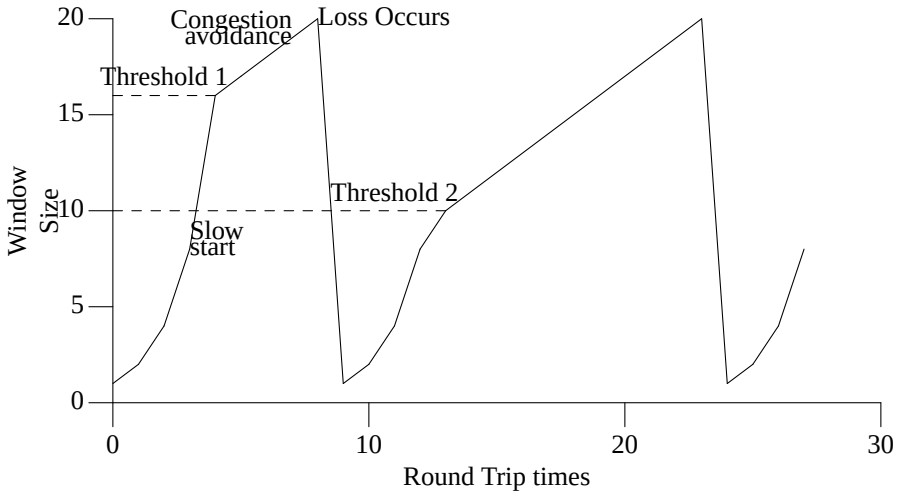
- Simple model: a source tracks x = the packets sent - the packets received
  As long as x < W the source can transmit the next packet
- d= round trip delay, X=packet transmission time
  - As long as $d \leq WX$ the source transmits continuously
  - When $d > WX$ the source must wait for permits
  - The maximum packet transmission rate is $r = \min\left(\frac{1}{X}, \frac{W}{d}\right)$

5.2.1.2 TCP

(A specific End-to-End window flow control protocol)

1. Advertised window
   - Receiver sends the source the number of bytes available in its buffer
   - Prevents buffer overflow at the receiver
2. Congestion window
   - Max number of bytes that the source can send without congesting the network
   - The congestion window starts at a low value and increases as long as the packets get through successfully
     ACK's are received
   - If a packet is lost, the window size is reduced to the minimum

— The maximum number of bytes that the source can send is the minimum of the advertised window and the congestion window



3. Regions of operation

A. Slow Start

- initially set window to 1 packet
- send the packet and wait for an ACK
- Each time an acknowledgment is received, increase the number of packets that can be outstanding by 1.
- in the initial phase, one packet is sent.
- in the second phase, one ack is received, and 2 packets are transmitted
- in the third phase, two acks are received, and 4 packets are transmitted
- in the fourth phase, 4 acks are received, and 8 packets are transmitted.
- The number of packets transmitted increases exponentially in each phase, as long as the network doesn’t lose packets.

B. Congestion avaoidance

- Initially, the slow start bound, "congestion threshold" is set to 65,635 bytes
- When the slow start phase reaches the bound, slow down the rate of increase
- instead of increasing the number of packets that can be outstanding by 1 for each ack received, increase the number of outstanding packets by 1 for each phase.
- If slow start ends at 16 outstanding packets, when the 16 acks are received increase the number of outstanding packet to 17.
- When the 17 acks are received, increase the number of outstanding packets to 18, and so on, untils a packet is lost.
  or we reach the maximum advertised window.

C. Congestion

- When an ack is not received, indicating that a packet was lost, or if a duplicate ack is received, indicating a lost packet that is retransmitted, or an out of order packet, we assume that the cause is network congestion that has caused a buffer overflow, rather than a transmission error. This assumption is valid in fiber optic networks, but not in wireless networks.
- the "congestion threshold" is set to the max ((1/2 of the current window), 2)
- in the example, the first packet is lost when the window is 20, so the congestion threshold is set to 10
- the window size is reduced to 1, and slow start is used again
- note that the congestion protocol does not work if many messages are lost because of transmission errors, as may happen in wireless networks.

4. Determining when a packet is lost

   A. Estimate the average round trip time, $\overline{RTT}$, as the running average
   $\overline{RTT} \leftarrow \alpha\overline{RTT} + (1-\alpha)RTT$
   Where RTT is the last measured time from message transmission to the reception of an ACK

   B. Estimate the mean deviation, DEV, from the average as:
   $DEV \leftarrow (1-\delta)DEV + \delta|RTT - \overline{RTT}|$

   C. The upper bound on the allowable round trip time is calculated as:
   $RTO = \overline{RTT} + 4 * DEV$

   D. When the delay is > RTO, the packet is assumed to be lost

5. This is the standard TCP algorithm, more recently TCP Tahoe and TCP Reno have been proposed to change the startup and backoff procedures

5.2.1.3 Hop-by-Hop windows

Each node has a fixed number of buffers for a path.

When the buffer is full permits are not returned

Instead of packets filling the network, a source that uses a congested link is slowed down - back pressure
This a generalization of the Beeforth protocol

5.2.2 Isarithmic Flow control

- Limit the total number of permits in the network, rather than the tokens on a specific path
- No way to keep all of the tokens from applying to a congested area of the network
- Difficult to recover lost permits

5.2.3 ATM Networks

— Congestion control is used for available bit rate sources (ABR)

— The objective is to give ABR sources as much bandwidth as possible after the contracts with constant bit rate source (CBR) and variable bit rate sources (VBR) have been satistfied.

— Every $N_{RM} - 1$ cells an ATM source sends a resource management cell (RM)

— The RM collects information at the intermediate switches, and the destination turns the RM around and sends it to the source.

— The source uses the information in the RM to control its transmssion rate.

— the information collected by RM varies with the application

— Binary Feedback

- RM cells have a congestion indication bit, CI, which is initially zero
- a congested switch, with long queues, sets a local Explicit Feedback Congestion Indicator, EFCI, to 1
- While EFCI is 1, the destination sets the CI bit to 1
- If the source receives RM's with CI=1, or does not receive RM, indicating that they were lost, it decreases its rate
- If the source receives RM's with CI=0, it can increase its rate.

— Explicit rate feedback

- A source puts its desired rate in the RM
- Any switch may lower the rate in RM
- The rate returned to the source is the min rate on the path
- A switch calculates the available rate as B/n, where B is the bandwidth it has available for ABR sources and n is the current number of ABR circuits in the switch
- This is a rough approximation of bottleneck flow control, but doesn't consider flows through a switch that are saturated elsewhere in the network, and cannot use the assigned rate.

— Enhanced proportional rate control algorithm (EPRCA)

- use both binary feedback and explicit rate feedback so that cells can pass through more complex switches and simpler switches

6. Models for flow control in wireless networks

6.1 Problems that are unique to wireless

In a wired network all of the flows that share a channel access the channel at the same source node. Each flow can be assigned a fraction of the capacity that is considered to be its fair share.

In a wireless network:

I. The flows that share the channel may originate at different nodes that may 1) be prevented from transmitting by the source, or 2) may interfere with the transmission from the source at the reciever.
In an 802.11 network

1. the source sends an RTS signal. The RTS inhibits any nodes that receive the signal from a) agreeing to receive a signal from another source or b) transmitting a signal. Part b is not necessary. As long as the sources that receive a RTS do not receive a CTS from the receiver, they will not interfere at the receiver.

2. The intended receiver sends a CTS signal. The CTS inhibits nodes that receive the signal from a) agreeing to receive a signal from another source, or b) transmitting a signal. Part a) isn't necessary because if the node doesn't receive an RTS from the source, a source that is not inhibited from transmitting will not interfere at the receiver.

Transmission of a packet involves contention with the receivers in the neighborhood of the source and transmitters in the neighborhood of the receivers. The level of contention for the shared wireless channel in a geographical region is dependent on the number of contending nodes in the region.

This is fundamentally different from wireline and cellular channel models, wherein all flows perceive the same contention. The nodes that are blocked from initiating or accepting a transmission may inhibit a different set of flows from transmitting or receiving. Therefore, it is not obvious which set of flows share a channel.

II. Trade-off between channel utilization and fairness:
In a a multi-hop wireless networks, the communications bandwidth in a region is shared. While spatial reuse is very useful for increasing the utilization of the wireless channel, it introduces a fundamental conflict between optimizing aggregate allocated bandwidth and achieving fairness, because allocating the channel to a flow with a large contention correspondingly reduces the channel reuse. In contrast, wireline and cellular networks do not face this problem because all flows perceive the same contention.

III. In a wireline network the node that transmits on a channel has information about, and controls, all of the flows that use the channel. In a wireless network the flows that share the ether originate at different nodes that may not have accurate information on all of the flows that are available to transmit on the channel at any instance.

6.2 Cliques - A Micro-model

A graph of cliques are used to determine which sources can transmit at the same time and to assign rates to the sources that share a transmission region.

To determine cliques we

1. Draw a graph that shows which nodes are within the transmission range of a source

2. Consider only the nodes with packets to transmit, and draw a graph that shows which packets will interfere with each other.

3. Draw a bipartite graph that shows which nodes can transmit simultaneously.

Consider a simple network with 5 nodes in a linear arrangement. The dotted ellipse around a node shows

the other nodes that are within the nodes transmission range. The solid lines show the connectivity of the network.

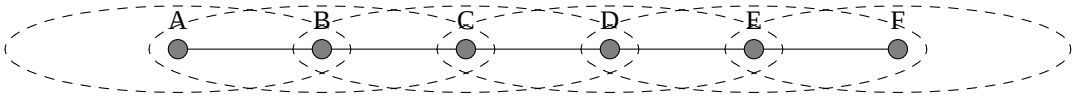


Figure 1. Network graph. Neighbors that interfer.

The second step is to draw a transmission contention graph. This is the set of transmissions that interfere with each transmission. In this step we only consider node pairs in the network graph that are actively transfering packets. Each node in this graph represents a transmission between a node pair in the previous graph that is using the channel.

For instance, assume that there are active transmissions between every pair of connected nodes in the previous graph. There are active transmissions between nodes AB, BA, BC, CB, DE, ED, EF, FE. The graph has lines between transmissions that interfere with one another at the source or destination nodes.

The transmissions from CD and DC have the same interference and are drawn as a single node. The nodes that C prevents from receiving with when it transmits are the same nodes that interfer with C when C is receiving. Similarly, the nodes D prevents from receiving when it transmits, prevent it from receiving when they transmit. Therefore, the set of nodes that can transmit simultaeouly with CD are the same as the set of nodes that can transmit simultaneously with DC.

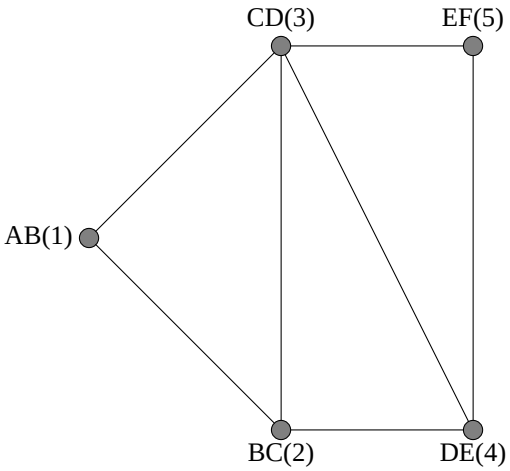


Figure 2. Transmission Contention Graph

The transmission contention graph shows which of nodes can or cannot transmit simultaneously. For instance, 1 and 5 can transmit simultaneously, because there isn't an edge connecting them, while 1 and 3 cannot. A clique is a set of nodes that are all within distance 1 of one another and cannot transmit simultaneously. A clique is a maximal clique if we cannot add any more nodes to the clique without having two nodes in the set that can transmit simultaneously. nodes 1 2, and 1 3 form two cliques. Nodes 1 2 3 form a maximal clique. If we try adding node 4 to the clique, nodes 1 and 4 can transmit simultaneously, and if we add node 5 to the clique, nodes 1 5 or 2 5 can transmit simultaneously. Nodes can transmit simultaneously when their distance is 2 or greater.

The maximal cliques define contention regions. At most one node in each maximal clique can transmit simultaeously. The contention regions in the transmission contention graph are (1 2 3), (2 3 4), and (3 4 5).

In step 3 we draw a bipartite graph with the active transmissions on one face and the contention regions on the other. The edges show which active transmissions are in each contention region.

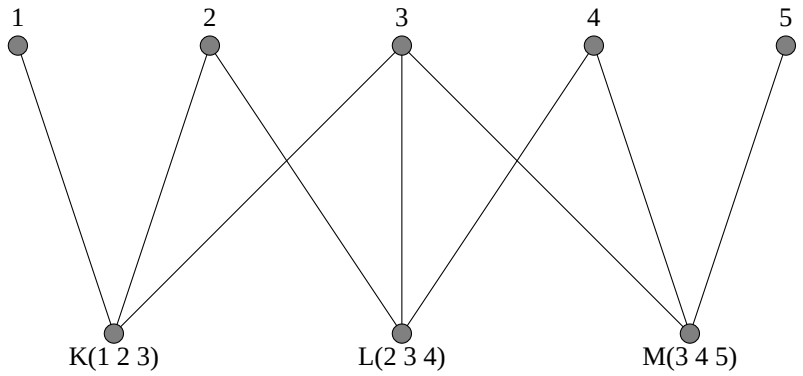


Figure 3. Resource Contention Graph

The contention regions are resources. Only one of the transmission in a contention region can occur simultaneously. However, in order for a transmission to be successful, it must simultaneously be assigned the transmission rights in all of its transmission regions. Transmission 1 is successful if it obtains the transmission rights in region K, while Transmission 4 must simultaneously obtain the transmission rights in regions L and M.

Note that a flow from node A to F would cause transmissions AB, BC, CD, DE, and EF (1, 2, 3, 4, 5). Packets from the same flow interfer with one another. In this example, each packet in the flow contends with two other packets in the same flow in each of the resource regions. If we allocate capacity based on the system resources that are consumed, some flows can requires several times as much of the bandwidth in a region.

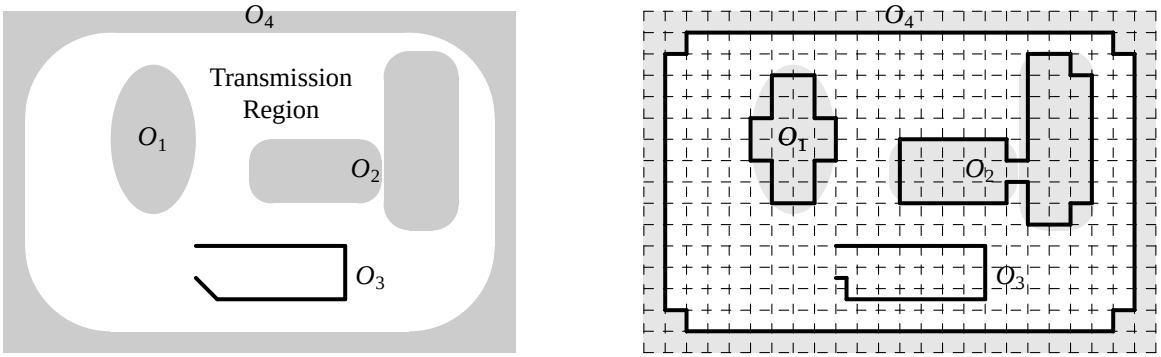
6.3 Clustering - A Macro Model:

- Divide nodes into regions with groups of nodes that are near one another, instead of treating each node individually
- Hierarchical structure allows routing and flow control algorithms to be applied to a larger number of nodes
- Effect of Node Density

  With a large number of nodes the regions with nodes remains stable

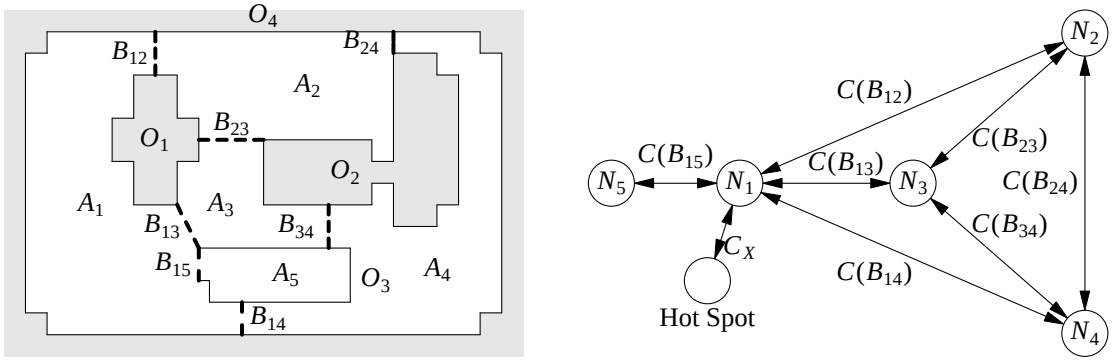
  New Cluster heads must be elected when nodes leave a region.

Obstacle Based Model



a) Obstacles in the Transmission Region b) Approximating Obstacles by a grid

A network is comprised of obstacles and transmission regions



a) Transmission Regions and Bottlenecks b) Super-nodes and Undirected Links

Models of the Network

7. Buffer Management

7.1 Beeforth's protocol [12]

Objective: To retain packet sequence in a virtual circuit network when packets are lost on a path, and to limit the buffering at intermediate nodes

A protocol mechanism, rather than a routing discipline, such as hot-potato routing, is used to limit the amount of buffering in the network

This protocol is particularly useful in all optical networks. Some optical switches have been proposed that transfer signals similar to the double acks.

Implementation

1. first packet is the scout and reserves 1 buffer at each node
2. 2 - acks per link
3. A message is periodically retransmitted on a link until ACK 1 is received

   Ack 1 says that the packet was successfully transferred to the next node

   The node can stop transmitting the packet

   The transmitting node can discard the packet, since it has been successfully forwarded, and make the buffer available to receive another packet.
4. When the buffer becomes available for the next packet, a node sends the node that transmitted the packet ACK 2

   ACK 2 is an invitation to send the next packet in the message

   If ACK 2 is lost the previous node will not transmit the next packet

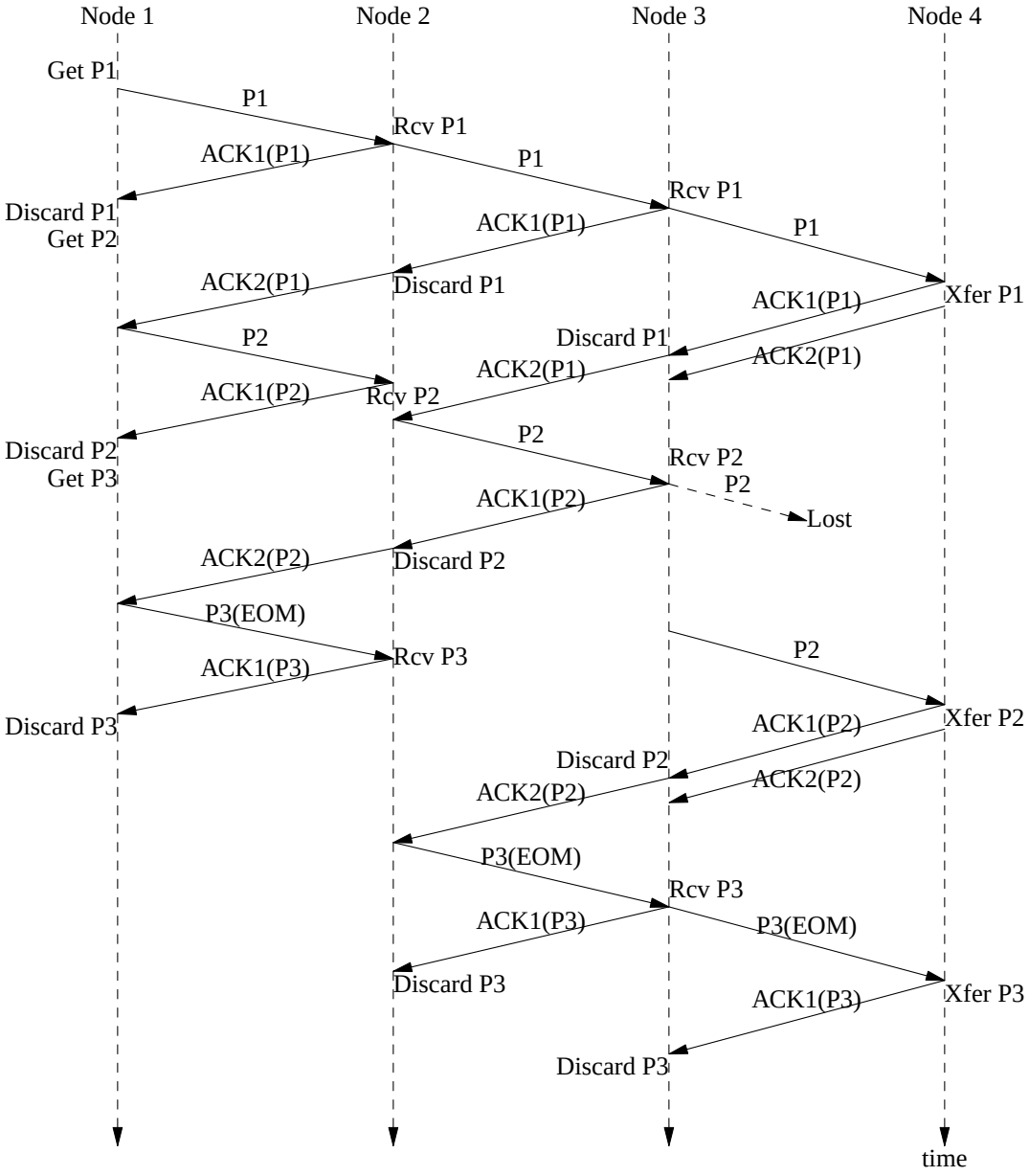
   ACK 2 is periodically retransmitted until the next packet is received

   The retransmit time of ACK 2 is longer than the retransmit time for messages, so that ACK 2's are not retranmitted when a message is lost
5. The last packet in a message contains an EOM

   When a node receives ACK 1 for the EOM, it does not send ACK2, since there are no more packets

   When ACK 1 is received, the buffer is deallocated

   If the final ACK 1 is lost, the node continues transmitting the packet until it is received


Node 1
Node 2
Node 3
Node 4
Get P1
P1
Rcv P1
ACK1(P1)
P1
Discard P1
Get P2
Rcv P1
ACK1(P1)
P1
ACK2(P1)
Discard P1
Xfer P1
ACK1(P1)
P2
Discard P1
ACK2(P1)
ACK2(P1)
ACK1(P2)
Rcv P2
Discard P2
Get P3
P2
Rcv P2
P2
ACK1(P2)
Lost
ACK2(P2)
Discard P2
P3(EOM)
P2
Rcv P3
ACK1(P3)
Discard P3
Xfer P2
ACK1(P2)
Discard P2
ACK2(P2)
ACK2(P2)
P3(EOM)
Rcv P3
ACK1(P3)
P3(EOM)
Discard P3
Xfer P3
ACK1(P3)
Discard P3
time

Homework

Beeforth's Protocol: Consider a path $N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow N_4$. Write the time sequence of messages on each of the links for packets P_{i-1} to packet P_{i+3} when

A. packet P_i is lost on link $N_2 \rightarrow N_3$

B. ACK_1 for packet P_i is lost on link $N_3 \rightarrow N_2$

C. ACK_2 for packet P_i is lost on link $N_3 \rightarrow N_2$

7.2 Deadlocks

References 13, 14, 15, 16
In Beeforth's protocol we allocate at least 1 buffer per flow at each node.
We can share the buffers between flows, but this can result in deadlocks

Problem:

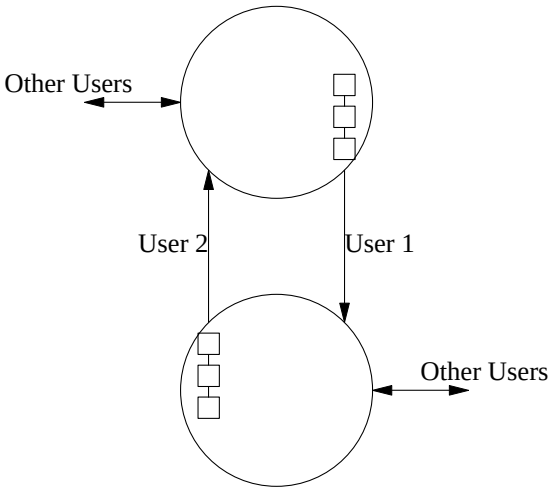
- We acquire and hold resources in a network, but the resources are not used until we acquire all of the resources on a path, or complete transmission along a path
- Holding the resources prevents others from acquiring the resources
- Several users holding resources prevent each other from completing a path

Examples

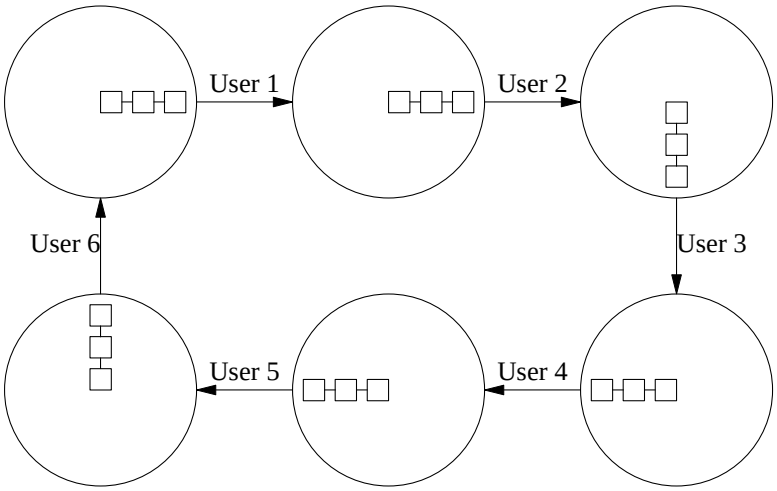
1. Message holds a buffer until a buffer at the next node is acquired
2. Bandwidth is acquired on a link on a path and held until a source-destination path is complete

Types of Deadlocks

1. Direct deadlock
   User 1 blocks user 2, while user 2 blocks user 1



2. Indirect Deadlock
   A cycle of users prevent each other from progressing



A solution:

Merlin and Schweitzer

- M = longest path from a source to a destination
- Divide the buffers at each node into M+1 buffer pools, labelled 0, 1, $\cdots$, M each with 1 or more buffers.
- **Rule:**
  1. A packet from a source can only enter the network when there is an empty buffer in the buffer pool at level 0
  2. A packet in a buffer pool at level i at node n_A, can only progress to node n_B when there is an empty buffer in the buffer pool at level $i + 1$ at n_B.
  3. A packet at the destination will always leave the network

Proof that there are no deadlocks

1. A packet in a buffer at level M can always leave the network.
   - The packet must be at the destination, since the maximum path length is M.
   - Packets at their destination can always leave the network
2. A packet in a buffer at level M-1 can always progress.
   - If the path length is M-1, the packet is at the destination and can leave the network.
   - If the path length is M, the buffer at level M at the next node will leave the network, and that buffer will become available.
3. By iteration from i=M-2 to i=0, a packet in a buffer at level i can always progress.
   - If the path length is i-1, the packet is at the destination and can leave the network.
   - If the path length $\geq i$, the packet in the buffer pool at level $i + 1$ at the next node will progress, and the buffer will become available.

4. Packets from a source can always enter the network.

   • a packet in a buffer at level 0 will always progress, and that buffer will become available.

Problems:

1. A packet from a source may have to wait to enter the network even though there is an empty buffer at the node.

2. A packet may have to wait to be forwarded even though there is an empty buffer at the next node.

Modifications:

1. Increase the buffers that we can use when we enter the network.
   A packet from a source with a path length K to the destination, where $1 \le K \le M$ can enter the network in any buffer pool at level L, where $0 \le L \le M - K$.

   • The packet will reach the destination and leave the network before its buffer pool level reaches M+1

2. Increase the buffers we can use in an intermediate node by using any lower numbered buffer
   A packet at level i at node n_A can be forwarded to any empty buffer at level $L \le i + 1$ at n_B.

   • All packets at level $L \le i + 1$ will reach the destination and leave the network before its buffer pool level reaches M+1

3. Increase the buffers we can use in an intermediate node by using some higher numbered buffers.
   A packet at level i at node n_A, that is at distance K_d from its destination, can be forwarded to any buffer in node n_B at level $L \le M + 1 - K_d$, even though $L > i + 1$.

   • Mod 3 compliments Mod 2, because packets whose level has been reduced, can later be raised

This is not a complete solution. Livelocks

- There may be packets at level i at nodes n_B and n_C waiting for a buffer at node n_A
- If the packet from node n_B is always forwarded first, another packet at level i may arrive at n_B before another buffer becomes available at n_A.
- A *livelock* can exist where node n_C is blocked by a stream of packets from node n_B
  This is a livelock rather than a deadlock because data continues to flow through the network.
- Livelock can be prevented by doing round robin service of the nodes waiting for a buffer.
- Livelock can also be prevented by giving priority to the packets that have been in the network for the longest time.

The problem of buffer allocation and bandwidth management are analagous

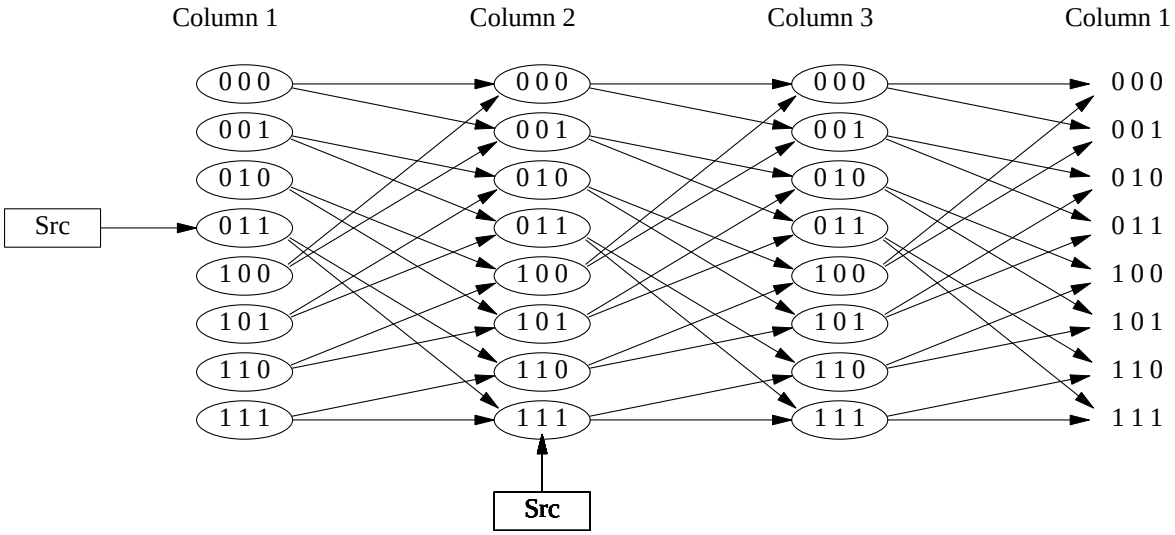
- Assume that a circuit acquires capacity on a link and holds that capacity until it acquires a complete path to the destination.
- Once a circuit acquires a complete path, it transmits data, and eventually releases all of the acquired capacity
- Divide the capacity on a link into pools, the same as the buffer levels.
- Eventually all circuits are acquired - there is no deadlocks where circuits that are holding partial paths block each other forever.

Homework

Deadlock Avoidance

Consider the 24 node cyclotomic shuffle-exchange network with fixed size cells:

- There is a source and destination at each node.
- A cell cannot be forwarded to the next node until a buffer is available



A. What is the minimum number of buffers needed at each node to guarantee that there are no deadlocks?

REFERENCES

[1] D. Bertsekas, R. G. Gallager, **Data Networks**, Prentice-Hall Inc, 1992.

[2] M. Gerla, H. W. Chan, "Fairness in Computer Networks," ICC-85, Chicago, June 1985, pp. 1384-1389.

[3] R. G. Gallager, S. J. Golestani, "Flow Control and Routing Algorithms for Data Networks," Proc. Int. Conf. on Computer Commun., Atlanta, GA, Oct. 1980, pp.779-784

[4] L. Massoulie, J. Roberts, "Bandwidth sharing: objectives and algorithms," IEEE/ACM Trans. on Networking, June 2002, pp. 320-328.

[5] JEAN-YVES LE BOUDEC, "Rate adaptation, Congestion Control and Fairness: A Tutorial," Ecole Polytechnique FÂ´edÂ´erale de Lausanne (EPFL), December 9, 2008,

[6] T. Nandagopal, Tae-Eun Kim, Xia Gao, V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks," ACM MOBICOM 2000, Boston MA USA, pp 87-98.

[7] Frank Kelly, "Charging and rate control for elastic traffic," European Transactions on Telecommunications, Vol. 8, Iss. 1, pp. 33-37, Jan.-Feb. 1997.

[8] J. Jaffe, "Bottleneck Flow Control", IEEE Trans. on Comm., Vol. Com-29, No. 7, July 1981, pp. 954-962.

[9] M. Gerla, M. Staskauskas, "Fairness in Flow Controlled Networks," ICC 1981, pp. 63.2.1-5

[10] Congzhou Zhou, N. F. Maxemchuk, "Bandwidth Balancing in Mobile Ad Hoc Networks," IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob'2010, Oct. 11-13 Niagara Falls, Canada.

[11] A. Leon-Garcia, I. Widjaja "Communication Networks: Fundamental Concepts and Key Architectures," McGraw-Hill, 2000.

[12] T. H. Beeforth, R. L. Grimsdale, F. Halsall, D. J. Woolons, "Proposed Organisation for Packet-Switched Data-Communication Network," Proc. IEE, vol. 119, no. 12, Dec 1972, pp. 1677-1682.

[13] P. M. Merlin, P. J. Schweitzer, "Deadlock Avoidance - Store-and-Forward Deadlock," IEEE Trans. Commun., vol COM-28, pp. 345-354, Mar. 1980.

[14] P. M. Merlin, P. J. Schweitzer, "Deadlock Avoidance in Store-and-Forward Networks II - Other Deadlock Types," IEEE Trans. Commun., vol COM-28, pp. 355-360, Mar. 1980.

[15] Y. B. Ko, N. H. Vaidya, "Location-Aided Routing (LAR) in mobile ad hoc networks," Wireless Networks, 2000, v6, n4, pp 307-321.

[16] M. Karol, J. Golestani, D. Lee, "Prevention of Deadlocks and Livelocks in Lossless Backpressured Packet Networks," IEEE/ACM Trans. on Networking. vol. 11, no. 6, Dec. 2003, pp. 923 - 934.