A Flexible and Adaptive Admission Control Framework for DiffServ Access Networks

jani.lakkakorpi@iki.fi

Nokia Research Center / Communication Systems

8 January, 2004

IRoNet Results Seminar

Disclaimer

“I do not know (yet) how the scheme I am going to present relates to IRoNet activities. This is just something I have been doing for the past two years – mostly for my Licentiate thesis.”

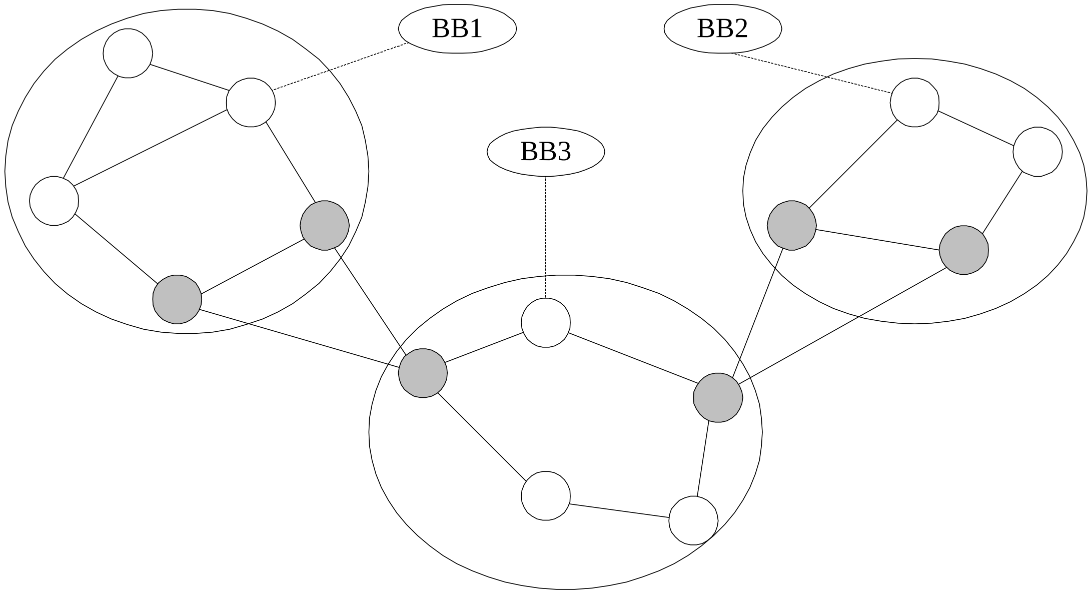
Agenda

- Motivation for flexible connection admission control
- Bandwidth Broker framework
- Introducing measurements to Bandwidth Broker framework
- Flexible and adaptive connection admission control
  - Flexible bandwidth sharing between admission-controlled traffic classes
  - Pricing can be taken into account in admission decisions
  - Adaptive AF weight tuning
  - Adaptive reservation limit tuning
- Performance evaluation (simulation results)
- Conclusions & future work

Why Flexible Admission Control?

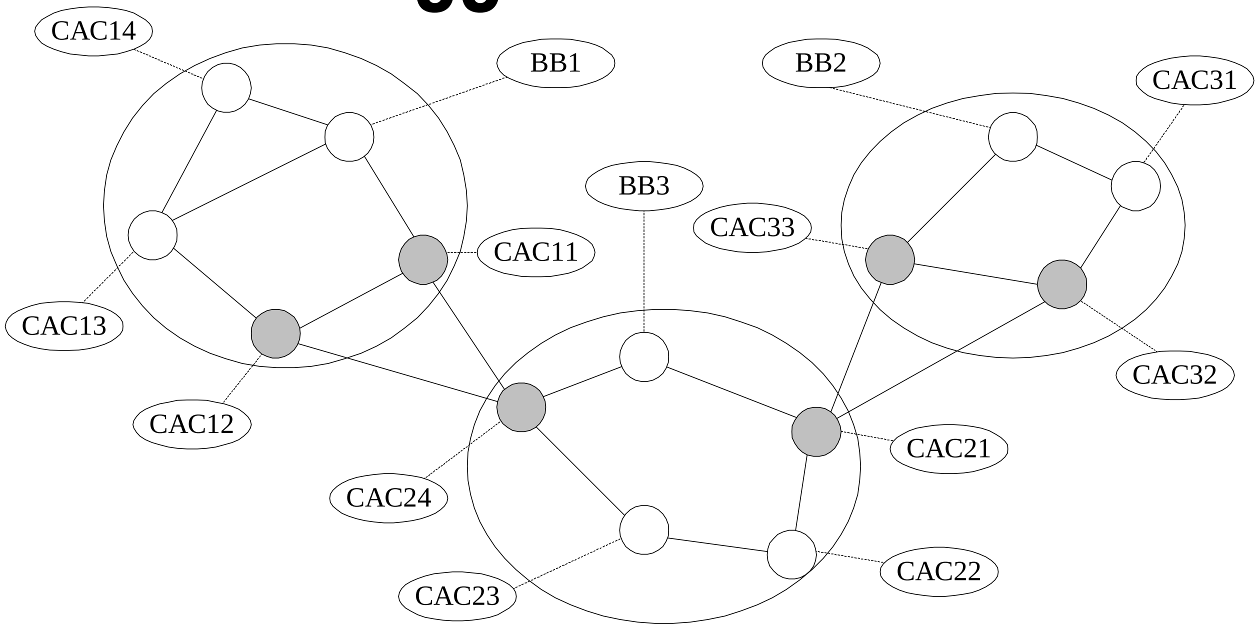
- Why admission control in IP networks at all?
  - There are applications (e.g., VoIP) that don't adapt to network congestion.
  - Overprovisioning? OK, but how much is really enough?
  - DiffServ reduces the need of new capacity but it does not create bandwidth out of thin air.
- Why different kind of admission control rules for different type of connections? Isn't one threshold for all connections under admission control enough?
  - Thanks to DiffServ, the requested bandwidth does not have to be equally guaranteed for all connections (e.g., VoIP vs. streaming).
  - Hard vs. soft guarantees can be realized with the help of differentiated edge policing and packet forwarding in the core.
- Why not to use the application requirements as a basis for admission control decisions?
  - This would lead into favoring real time applications, which may not be always beneficial.

Bandwidth Broker Concept



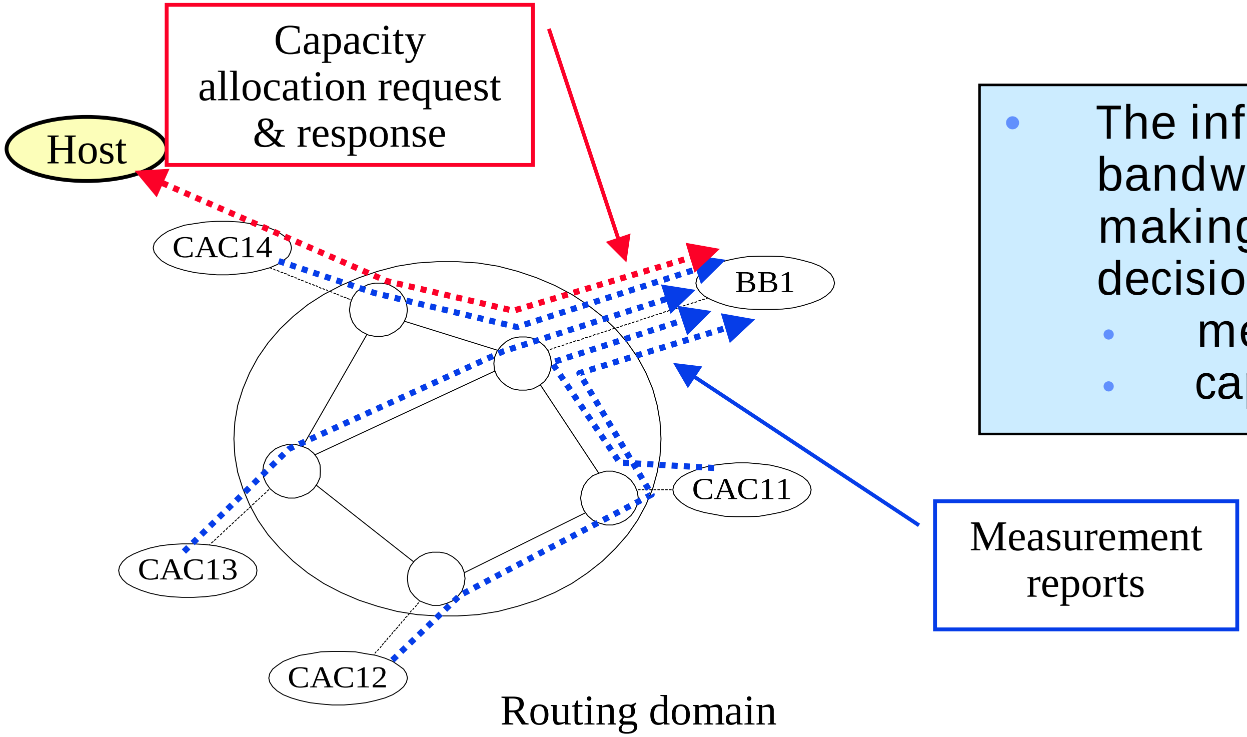
- RFC 2638 (A Two-bit Differentiated Services Architecture for the Internet) introduced a Bandwidth Broker (BB) agent that has the information of all resources in a specific domain.
- O. Schelén has presented an admission control scheme, where clients can make bandwidth reservations through BB agents:
  - For each routing domain, there is a BB agent responsible for admission control.
  - BB agent maintains information about reserved resources on each link in its routing domain.
  - BB learns the domain topology by listening to OSPF messages and link bandwidths through Simple Network Management Protocol (SNMP).
  - BBs are responsible for setting up policers at the network edges.

Suggested Additions



- The use of static reservations only (parameter-based admission control, PBAC) can leave the network underutilized. This is due to the fact that average bit rates can be substantially lower than the corresponding (requested) peak rates. Link load measurements are needed for more efficient network utilization.
- CAC agents monitor and update their “local link loads” by using exponential averaging on their local router statistics. CAC agents also send periodical load updates to BB agent of the routing domain.

Distributing the Information



- The information that a bandwidth broker needs for making admission control decisions is retrieved through:
  - measurement reports
  - capacity allocation requests

CAC agents send **exponentially averaged** link load information to BB every p seconds:

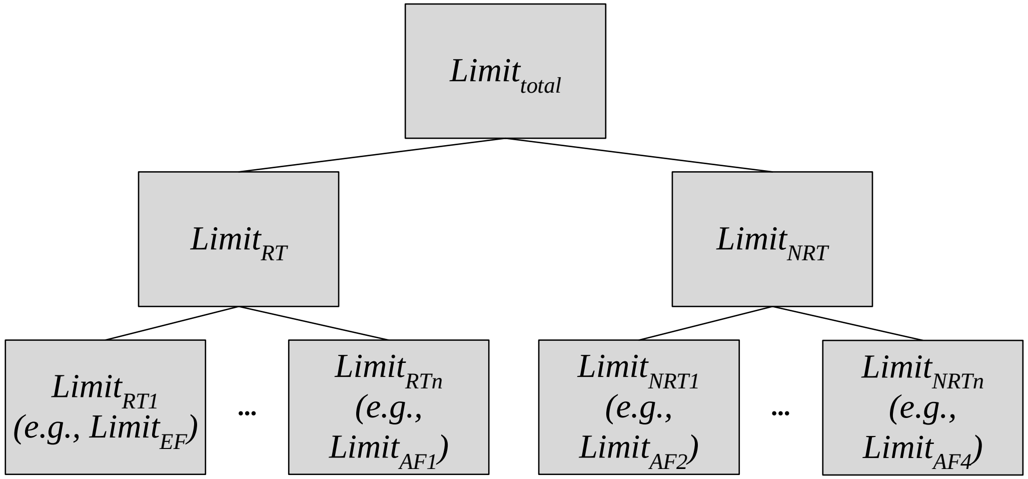
$$load_{class} := (1-w) * load_{class} + w * currentLoad_{class}$$

$$currentLoad_{class} := \max(\frac{dequeuedBits_{class}(1)}{s * bw}, \dots, \frac{dequeuedBits_{class}(p/s)}{s * bw})$$

s is the sampling period, p is the measurement period and w is the averaging weight.

Load & Reservation Limit Hierarchy:

Available Path Bandwidth Calculation



$Limit_{total}$

$Limit_{RT}$

$Limit_{NRT}$

$Limit_{RT1}$ (e.g., $Limit_{EF}$)

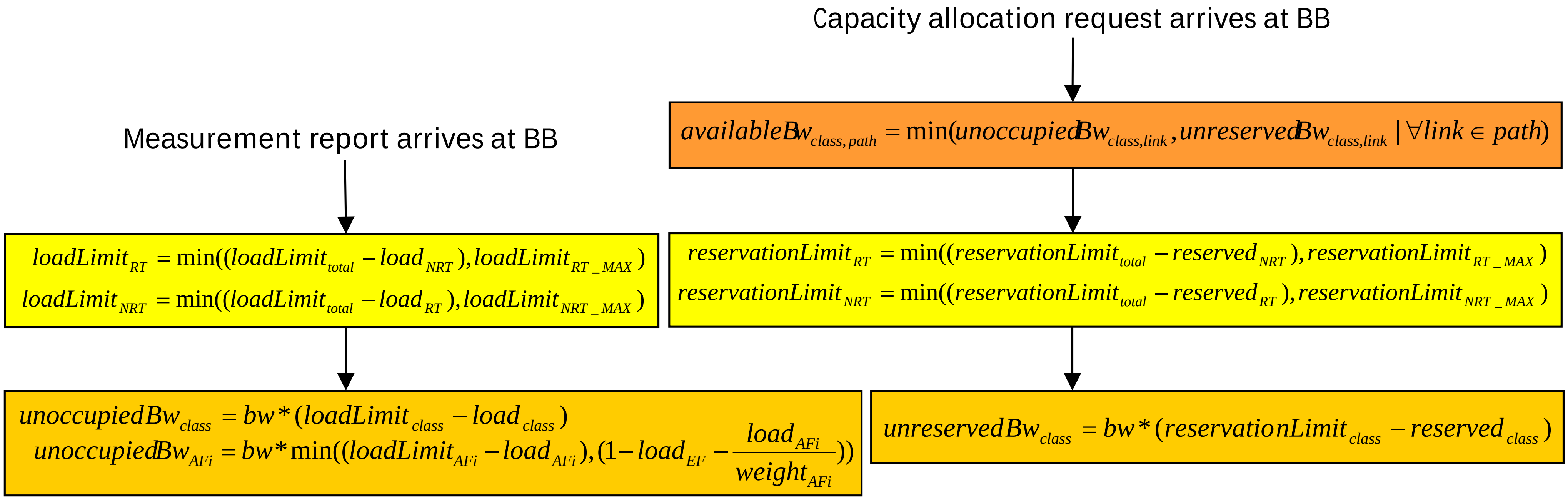
...

$Limit_{RTn}$ (e.g., $Limit_{AF1}$)

$Limit_{NRT1}$ (e.g., $Limit_{AF2}$)

...

$Limit_{NRTn}$ (e.g., $Limit_{AF4}$)



Capacity allocation request arrives at BB

$$availableBw_{class,path} = \min(unoccupiedBw_{class,link}, unreservedBw_{class,link} \mid \forall link \in path)$$

Measurement report arrives at BB

$$loadLimit_{RT} = \min((loadLimit_{total} - load_{NRT}), loadLimit_{RT\_MAX})$$
$$loadLimit_{NRT} = \min((loadLimit_{total} - load_{RT}), loadLimit_{NRT\_MAX})$$

$$reservationLimit_{RT} = \min((reservationLimit_{total} - reserved_{NRT}), reservationLimit_{RT\_MAX})$$
$$reservationLimit_{NRT} = \min((reservationLimit_{total} - reserved_{RT}), reservationLimit_{NRT\_MAX})$$

$$unoccupiedBw_{class} = bw * (loadLimit_{class} - load_{class})$$
$$unoccupiedBw_{AFi} = bw * \min((loadLimit_{AFi} - load_{AFi}), (1 - load_{EF} - \frac{load_{AFi}}{weight_{AFi}}))$$

$$unreservedBw_{class} = bw * (reservationLimit_{class} - reserved_{class})$$

A Flexible CAC Instance with Three Classes:

Admission Decisions for EF, AF1 (RT) and AF2 (NRT) Connections

Bandwidth Broker:

```
for each admission request:
  classify connection ($class$ = EF/AF1/AF2)
  $admit$ = true
  if ($class$ != AF2)
    calculate $availableBw_{class, path}$ and $availableBw_{RT, path}$
    if (($availableBw_{class, path}$ < $f(price)$*$requestedRate$) OR ($availableBw_{RT, path}$ < $f(price)$*$requestedRate$))
        $admit$ = false
  else
    calculate $availableBw_{NRT, path}$
    if ($availableBw_{NRT, path}$ < $f(price)$*$requestedRate$)
        $admit$ = false
  if ($admit$ == true)
    for all links on the path:
      $reserved_{class}$ =+ $requestedRate$
      re-calculate $unreservedBw_{class}$, $unreservedBw_{RT}$, $unreservedBw_{NRT}$

for each connection tear-down:
  classify connection ($class$ = EF/AF1/AF2)
  for all links on the path:
    $reserved_{class}$ =- $requestedRate$
    re-calculate $unreservedBw_{class}$, $unreservedBw_{RT}$, $unreservedBw_{NRT}$

for each load update arrival:
  update link database: re-calculate $unoccupiedBw$:s
```

The effect of pricing: a function of the price the user is paying for the connection.

All CAC agents (including Bandwidth Broker):

```
timer expires:
  update link loads
  send update to Bandwidth Broker
  set timer to expire after $p$ seconds
```

Adaptive AF Weight Tuning (1/2)

- Motivation for AF weight tuning: If we give our “Best Effort” class a fair share of forwarding resources, say 10%, it is impossible to have strict priority like weights (e.g., 90:9:1) for the AF classes. Moreover, static “normal” AF weights could result into low bottleneck link utilization.
- Implementation:
  - Bandwidth Broker stores the $minUnoccupiedBw_{AFi}$ values for each link and performs periodical checks (every T_W seconds).
  - If certain thresholds are reached, new AF weights are applied for the involved links – and for the admission control algorithm.
    - If $minUnoccupiedBw_{AFi}/bw$ < $lowThreshold$ or $minUnoccupiedBw_{AFi}/bw$ > $highThreshold$, update $weight_{AFi}$.

$$\boxed{weight_{AFi} = load_{AFi} / (1 - load_{EF} - unoccupied)}$$

    - $weight_{AFi}$ is pre-calculated after each load update arrival – but only if $unoccupiedBw_{AFi}$ < $minUnoccupiedBw_{AFi}$. Here $unoccupied$ denotes the amount of link capacity that we would like to be always available. A negative value of $unoccupiedBw_{AFi}$ will immediately trigger AF weight tuning.
    - The final AF weights depend on the number of AF classes (N), excluding the “Best Effort” class.

$$\boxed{weight_{AFi} := weight_{AFi} / (\sum_{j=1}^{N} weight_{AFj}) * (1 - weight_{BE})}$$

Adaptive AF Weight Tuning (2/2)

Bandwidth Broker:

```
for each load update arrival:
  for each link mentioned in the message:
    update = false
    for each AF class under CAC:
      if ($unoccupiedBw_{AFi}$ < $minUnoccupiedBw_{AFi}$)
        $minUnoccupiedBw_{AFi}$ = $unoccupiedBw_{AFi}$
        $weight_{AFi}$ := $load_{AFi}/(1$ - $load_{EF}$ - $unoccupied$)
      if ($unoccupiedBw_{AFi}$ < $0$)
        update = true
    if update
      for each AF class under CAC:
        $weight_{AFi}$ := $weight_{AFi}$/sum($weight_{AFj}$ $j$ = $1$:$N$)*($1$ - $weight_{BE}$)
        $minUnoccupiedBw_{AFi}$ = $bw$
        enforce the minimum and maximum AF weights

timer expires:
  for each link:
    update = false
    for each AF class under CAC:
      if (($minUnoccupiedBw_{AFi}/bw$ < $lowThreshold$)||
           ( $minUnoccupiedBw_{AFi}/bw$ > $highThreshold$))
        update = true
      $minUnoccupiedBw_{AFi}$ = $bw$
    if update
      for each AF class under CAC:
        $weight_{AFi}$ := $weight_{AFi}$/sum($weight_{AFj}$ $j$ = $1$:$N$)*($1$ - $weight_{BE}$)
        enforce the minimum and maximum AF weights
  set timer to expire after $T_W$ seconds
```

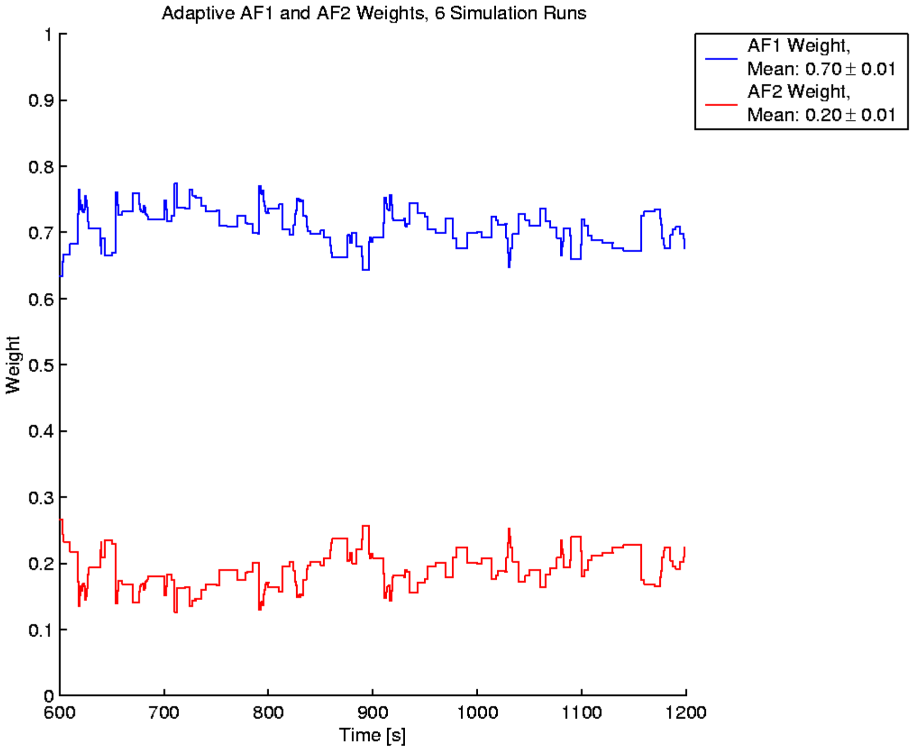
Adaptive Reservation Limit Tuning

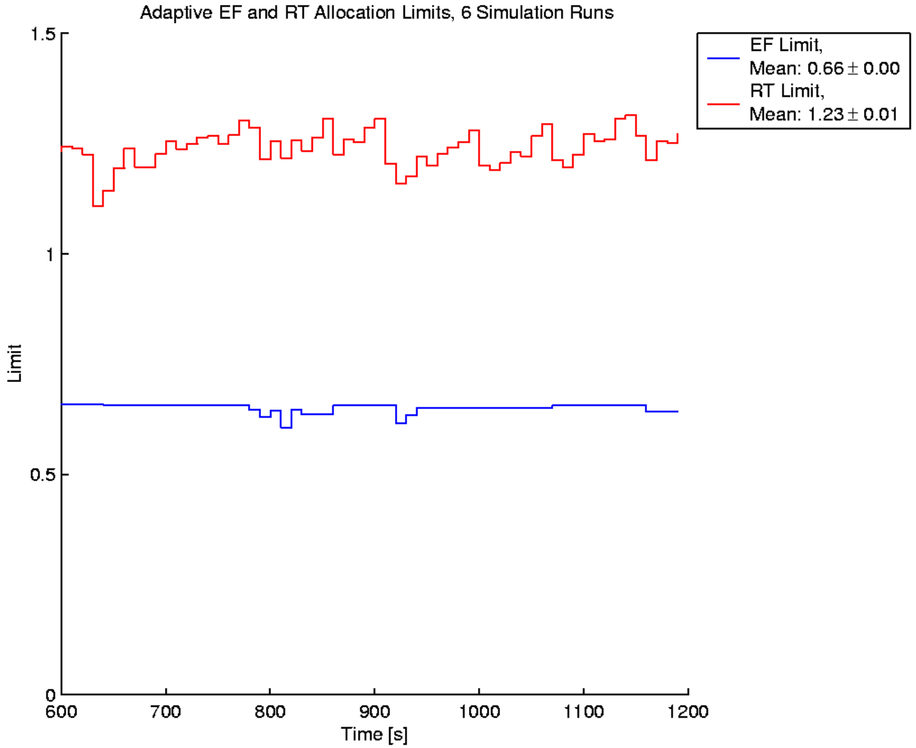
- Motivation for reservation limit tuning: Protection against a sudden burst of connection arrivals. Could be solved with strict PBAC but that would lead to low utilization.
- With adaptive reservation limit tuning, we effectively turn off the “real” PBAC. The MBAC part, however, does not change – connections can still be blocked because of exceeded link load thresholds.
- Reservation limit tuning for EF and RT classes:
  - Bandwidth Broker checks periodically (every T_R, e.g., 10 seconds) the $load_{EF}$ and $load_{RT}$ values of each link.
  - If the EF/RT reservation limit is too small compared to the actual link usage, we will increase the limit. Similarly, if the reservation limit is too big compared to the actual link usage, we will decrease the limit.
  - *increment* denotes the amount of capacity that we can increment to or decrement from the reservation limit.

Bandwidth Broker:

```
timer expires:
  for each link:
    if ($load_{EF}$ < ($loadLimit_{EF}$ - $increment$))
      $reservationLimit_{EF}$ = $reserved_{EF}$ + $increment$
    if ($load_{EF}$ > ($loadLimit_{EF}$ + $increment$))
      $reservationLimit_{EF}$ = $reserved_{EF}$ - $increment$
    if ($load_{RT}$ < ($loadLimit_{RT}$ - $increment$))
      $reservationLimit_{RT}$ = $reserved_{RT}$ + $increment$
    if ($load_{RT}$ > ($loadLimit_{RT}$ + $increment$))
      $reservationLimit_{RT}$ = $reserved_{RT}$ - $increment$
  set timer to expire after $T_R$ seconds
```

Adaptive AF Weights and Reservation Limits

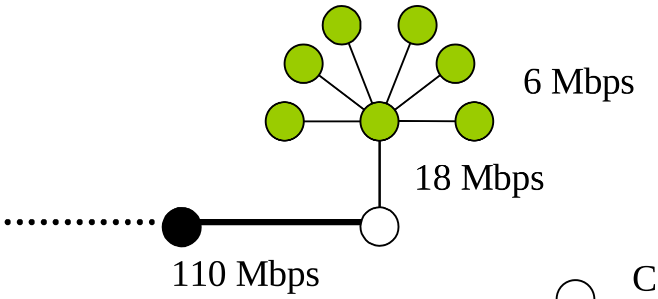




- Only the first of six simulation runs is graphed. Legend, however, provides the average values of all runs.
- AF3 class is given a static 10% weight.

Simulation Topology

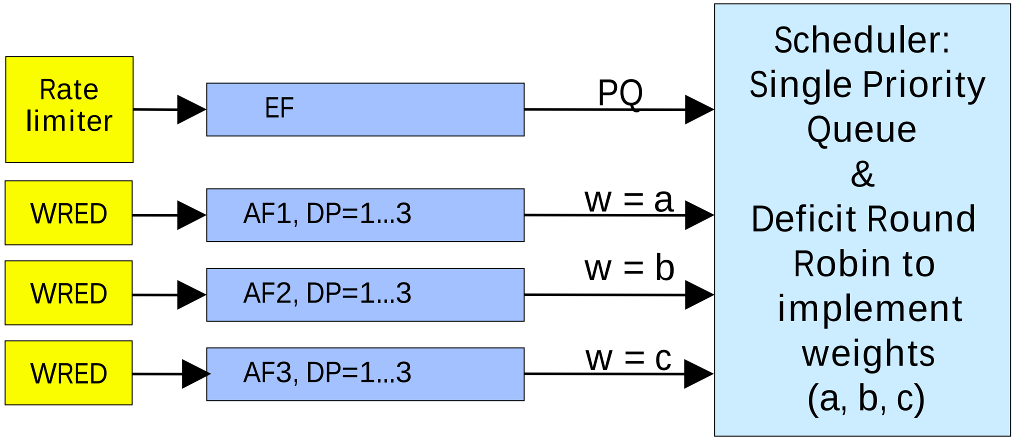
+Our EF/AF DiffServ Queueing Model + Traffic Mix + Service Mapping



○ Core router

● Edge router

● ”Access network gateway”


Rate limiter
EF
PQ
WRED
AF1, DP=1...3
w = a
WRED
AF2, DP=1...3
w = b
WRED
AF3, DP=1...3
w = c
Scheduler: Single Priority Queue & Deficit Round Robin to implement weights (a, b, c)

Service	Service level	PHB	Share of offered connections	Requested bandwidth (≈ peak rate)
VoIP calls	N/A	EF	20.0%	36 kbps
Videotelephony	N/A	EF	20.0%	84 kbps
Video streaming	Gold	AF11	4.0%	250 kbps
	Silver	AF12	4.0%	
	Bronze	AF13	4.0%	
Guaranteed web browsing	Gold	AF21	8.0%	250 kbps
	Silver	AF22	8.0%	
	Bronze	AF23	8.0%	
Normal web browsing and e-mail downloading	Gold	AF31	8.0%	N/A
	Silver	AF32	8.0%	
	Bronze	AF33	8.0%	

CAC Parameters

	No reservation limit tuning			EF and RT reservation limit tuning		
Parameters	SP like AF weights	Normal AF weights	Adaptive AF weights	SP like AF weights	Normal AF weights	Adaptive AF weights
$weight_{AF1}$	0.9	0.45	adaptive	0.9	0.45	adaptive
$weight_{AF2}$	0.09	0.45	adaptive	0.09	0.45	adaptive
$weight_{AF3/BE}$	0.01	0.1	0.1	0.01	0.1	0.1
T_W	N/A		10.0 s	N/A		10.0 s
lowThreshold	N/A		0.05	N/A		0.05
highThreshold	N/A		0.15	N/A		0.15
unoccupied	N/A		0.1	N/A		0.1
T_R	N/A			10.0 s		
increment	N/A			0.05		
$reservationLimit_{EF}$	10.0			adaptive		
$reservationLimit_{RT\_MAX}$	10.0			adaptive		
$reservationLimit_{AF1}$	10.0					
$reservationLimit_{AF2}$	10.0					
$reservationLimit_{NRT\_MAX}$	10.0					
$reservationLimit_{total}$	10.0					
$loadLimit_{EF}$	0.5					
$loadLimit_{AF1}$	0.5					
$loadLimit_{AF2}$	0.9					
$loadLimit_{RT\_MAX}$	0.9					
$loadLimit_{NRT\_MAX}$	0.9					
$loadLimit_{total}$	0.9					
$f(price)_{all}$	1.0					
s	500 ms					
p	1.0 s					
w	0.5					

Simulation Results

Method	EF+AF1+AF2 admission ratio [%]	Average EF+AF1+AF2 bottleneck load [%]	Maximum AF1 and AF2 delays [ms]	Maximum AF1 packet loss [%]
SP like AF weights (90:9:1), no tuning	37.6 ± 1.7	88.2 ± 0.1	4.9 ± 0.4 28.1 ± 14.8	1.4 ± 1.2
Normal AF weights (45:45:10), no tuning	45.2 ± 2.1	85.5 ± 0.5	9.4 ± 1.4 7.5 ± 0.7	7.8 ± 4.1
AF weight tuning	37.0 ± 2.2	88.1 ± 0.3	7.5 ± 0.6 28.4 ± 4.5	6.3 ± 2.0
SP like AF weights, EF & RT reservation limit tuning	41.4 ± 1.9	86.8 ± 0.2	4.0 ± 0.1 10.4 ± 0.7	0.1 ± 0.1
Normal AF weights, EF & RT reservation limit tuning	47.4 ± 1.7	84.7 ± 0.4	6.9 ± 0.7 7.2 ± 0.8	1.4 ± 0.4
AF weight and EF & RT reservation limit tuning	41.9 ± 1.8	86.7 ± 0.2	5.9 ± 0.2 12.6 ± 1.0	1.0 ± 0.4

- These results were obtained under bursty connection arrivals; created using a two-state Markov model. Arrival intensity: 5.83 1/s in the “normal state” and back-to-back arrivals in the “burst state”.
- Observations:
  - Adaptive AF weights result in good link utilization (better than with normal, non-SP, weights).
  - Reservation limit tuning results in significantly smaller packet loss ratio. Of course, this comes with a price of slightly lower link utilization.
  - AF weight and reservation limit tuning do not disturb each other.

Conclusions & Future Work

- In Flexible and Adaptive CAC, the demands of real time traffic do not override all other traffic but link resources are shared in a more flexible manner.
  - Price-based coefficients can be used in the admission decisions in order to maximize the total operator revenue.
  - AF scheduling weights can be tuned based on requested capacity and link load information.
  - Reservation limits can be tuned based on link load information.
- Simulations show that:
  - Measurement-based admission control is required also for EF traffic if high link utilization is desired.
  - Adaptive AF scheduling weights – in the CAC algorithm and in the routers – allow efficient use of available resources taking into account the QoS requirements of different class applications.
  - Adaptive reservation limits give good protection against bursty connection arrivals.
- Future/on-going work:
  - Make admission control aware of possible load balancing i.e. multiple paths need to be taken into account in admission decisions.

Thank You!