Ajil: Distributed Rate-limiting for Multicast Networks

11/19/08

1 Problem Specification

Let $P = \{n_1...n_N\}$ be the set of processes (nodes) in the system, let $g_j = \{n_i..\}$ be an multicast group with members $n_i \in P$, and let $G = \{g_1...g_M\}$ be the set of all multicast groups in the system. Let L be a global multicast rate limit set by the network administrator. Let $r^t_{g_j}$ be the multicast traffic rate of group g_j at time t. Let $s^t_{n_i,g_j}$ be the multicast sending rate of node n_i on group g_j at time t.

Our goal is to maintain that:

$$\forall t: \sum_{i=1}^{N}\sum_{j=1}^{M} s^t_{n_i,g_j} = \sum_{j=1}^{M} r^t_{g_j} \leq L$$

2 Trivial Approach

We can trivially achieve our goal by hand-wiring that

$$\begin{aligned} \forall t, \forall g_j \in G : r^t_{g_j} &= \frac{1}{M} \\ \forall t, \forall n_i \in P, \forall g_j \in G : s^t_{n_i,g_j} &= \begin{cases} \frac{1}{M \cdot |g_j|} & \text{if } n_i \in g_j \\ 0 & \text{if } n_i \notin g_j \end{cases} \end{aligned}$$

However, this quota allotment might not be fair to all the nodes and groups since some might require almost no sending quota while others might require much more. For this reason, we aim to design a protocol to assign sending quotas to nodes and groups based on their need and demand.

3 Our Approach

We build the `Ajil` protocol to assign multicast quotas for nodes and groups based on their needs and adapt to changes in their sending rates. On a high level this is done by dividing the time into equal-sized non-overlapping epochs (t) and applying a "slowdown policy" on a subset of the nodes if the global limit L has been violated in the previous epoch.

Our protocol lets nodes send multicast traffic without any restrictions. At the end of each epoch, each nodes independently evaluates whether the global limit L has been violated. If the limit has been exceeded, a subset of the nodes and groups is designated as a "Reaction Domain" for this epoch. The nodes in the reaction domain will apply a user-defined slowdown policy that dictates how their sending rates will be influenced in next epochs.

3.1 Broadcast Channel

We will use a broadcast channel to disseminate group aggregate traffic information (r_{g_j}) to all the nodes in the system. Upon receiving such an update r_{g_j}, a node will store it until another update about g_j is received, or for a finite timeout period after which the transmission rate on g_j is assumed to be 0. Let $B > 1$ be the maximum number of epochs a group traffic update is kept.

Given that there could be many members in each group, we would like to minimize traffic on the broadcast channel by having only one member of each group disseminate the rate aggregate information r_{g_j} at each epoch. As an added optimization, we would like to send these updates less frequently for groups that have very low traffic rates. Essentially we would like the broadcast periodicity to be set as: $p_{g_j} = a\frac{r_{g_j}}{L} + B$ where a is a constant. Thus for groups with $r_{g_j} = 0$, $p_{g_j} = B$. Similarly, for groups with high traffic

rates ($r_{g_j} = L$), we would like to have high broadcast rates with $p_{g_j} = 1$. This allows us to solve for $a + B = 1$, arriving at: $p_{g_j} = (1 - B)\frac{r_{g_j}}{L} + B$.

To avoid explicit coordination for broadcast duties, nodes can determine broadcast responsibilities probabilistically. At the end of each epoch, each node sends a broadcast update about its group g_j with probability:

$$\Pr_{broadcast} = \frac{1}{|g_j|} \cdot \frac{1}{|g_j|} \cdot \frac{1}{(1 - B)\frac{r_{g_j}}{L} + B} = \frac{L}{|g_j| \left((1 - B)r_{g_j} + L \cdot B\right)}$$

3.2 Reaction Domains

If the global rate limit is exceeded, each node decides whether to apply the slowdown policy or not by checking if it is in the reaction domain or not. A reaction domain (RD) is defined as the top β highest traffic nodes in the top α highest traffic groups. For this we explicitly define $0 < \alpha \leq 1$ to be the fraction of the highest traffic groups to be included in the RD, and $0 < \beta \leq 1$ to be the fraction of the highest traffic nodes to be included in the RD.

At the end of epoch t, $\forall i, j$ with $n_i \in g_j$ node n_i computes $\sum_j r^t_{g_j}$ based on the group aggregate information obtained from the broadcast channel. Assuming that $H \leq L$ is some threshold value, if $\sum_j r^t_{g_j} > H$ then n_i checks if g_j is in the reaction domain by computing if:

$$\texttt{traffic\_sorted\_index}(g_j) \leq \lceil \alpha M \rceil$$
(i.e. if g_j is one of the top α groups).

If so, n_i checks if:

$$\texttt{per\_group\_traffic\_sorted\_index}(n_i, g_j) \leq \lceil \beta |g_j| \rceil$$
(i.e. if n_j is one of the top β senders in group g_j).

If so, it knows that it is in the RD and applies the user defined slowdown policy.

3.3 Slowdown Policy

An administrator specifies a slowdown policy regarding what to do in case the global limit L is exceeded. The policy is applied to all the nodes in the reaction domain at the end of the epoch. The slowdown policy can be defined as any number of things for example:

- **Flat Tax:** each node slows its sending rate by some constant X KBps
- **Local Percentage:** each nodes $n_i \in g_j$ slows down by some percentage ω.
- **Global Percentage:** each node $n_i \in g_j$ slows down proportional to its local contribution to g_j and g_j's contribution to $\sum_j r^t_{g_j}$. This translates to slowing down by some percentage ω such that:

$$\omega = \frac{s^t_{n_j,g_j}}{r^t_{g_j}} \cdot \frac{r^t_{g_j}}{\sum_j r^t_{g_j}}$$

3.4 Rate Aggregation

Computing the broadcast probability and the reaction domain both require a node n_i to know the sending rates for other nodes in its groups and the aggregate rates for other groups it does not belong to. For that, each node n_i computes the sending rates for the other nodes in its groups locally as it receives their incoming messages. The sum of these sending rates $s^t_{n_i,g_j}$ constitutes the communication rate $r^t_{g_j}$ for the entire group at epoch t.

As some nodes may experience bursty behavior in their sending rates, each node keeps track of an exponential moving average (EMA) of the sending rates of other nodes in its group. More specifically, let $0 < \gamma \leq 1$ be the *smoothing factor* for the exponential moving average function. Each node $n_i \in g_j$ records

$$\forall n_k \neq n_i \in g_j : \; s^t_{n_k,g_j} = \gamma \cdot o^t_{n_k,g_j} + (1-\gamma) \cdot s^{t-1}_{n_k,g_j}$$

where $o^t_{n_k,g_j}$ is the observed sending rate of n_k in g_j during epoch t. Notice that this allows us to set $\gamma = 1$ and we will have $s^t_{n_k,g_j} = o^t_{n_k,g_j}$.

4 Algorithm Pseudocode

First, a list of all the variables and knobs in the system:

- $\boldsymbol{L}$: The global aggregate limit on all multicast traffic set on the network.
- $\boldsymbol{E}$: The length of an epoch.
- $\boldsymbol{B}$: The number of epochs before timing out a group's rate aggregate.
- $\boldsymbol{\alpha}$: The percentage of the highest traffic groups to include in the RD.
- $\boldsymbol{\beta}$: The percentage of the highest traffic nodes to include in the RD.
- $\boldsymbol{\gamma}$: The smoothing factor for traffic rate aggregates.

Algorithm 1 The Ajil Protocol for Multicast Rate-limiting

loop
  $o_{n_k,g_j} \leftarrow o_{n_k,g_j} + len(\text{packet from } n_k)$
  $r_{g_i} \leftarrow$ data from broadcast channel
  if $t \equiv 0(\mod E)$ **then**
    for all g_j, n_k **do**
      $s^t_{n_k,g_j} \leftarrow \gamma \cdot o^t_{n_k,g_j} + (1-\gamma) \cdot s^{t-1}_{n_k,g_j}$
      $r^t_{g_j} \leftarrow \sum_{n_k \in g_j} s^t_{n_k,g_j}$
    end for
    if $\text{rand}() \leq \frac{L}{|g_j|\left((1-B)r_{g_j}+L\cdot B\right)}$ **then**
      `broadcast`(r_{g_j})
    end if
    if $\sum_{g_j} r^t_{g_j} \geq L$ **then**
      sort groups by traffic
      if g_j's sorted index $\leq \lceil \alpha \cdot M \rceil$ **then**
        sort members of g_j by traffic
        if my sorted index $\leq \lceil \beta \cdot |g_j| \rceil$ **then**
          apply `slowdown` policy
        end if
      end if
    end if
  end if
end loop