The Role of Max-min Fairness in DOCSIS 3.0 Downstream Channel Bonding

Scott Moser, Jim Martin, James M. Westall, Brian C. Dean
School of Computing
Clemson University
Clemson, South Carolina
{smoser, jim.martin, westall, bcdean}@clemson.edu

***Abstract*—The concept of fairness in assigning access to network resources has been well defined in traditional networks. With the introduction of channel bonding into the DOCSIS 3.0 standard, where different flows may not have access to the same set of resources, the definition of fairness becomes less clear. In this paper we will investigate the application of max-min fairness in a channel bonded environment and implement an algorithm to allocate fair sharing of resources.**

Keywords- Channel Bonding; DOCSIS; Flow Networks; Max-min fairness; Scheduling

I. INTRODUCTION

Resource allocation is the process by which network capacity is apportioned among competing applications. A well designed resource allocation scheme for a network must demonstrate two crucial properties: efficient use of resources and an adequate level of user satisfaction. User satisfaction can be defined by metrics that quantify how well the network is meeting the user's service requirement. In situations where similar flows are competing for bandwidth, it is desired that the network fairly shares the available bandwidth.

Emerging broadband access technologies have unique properties that limit how readily existing definitions of fairness can be applied. In this paper we focus on DOCSIS 3.0 [1] HFC cable access networks where cable modems can receive data from multiple downstream channels and can send data over multiple upstream channels. DOCSIS 3.0 defines an abstraction called a bonding group to facilitate centralized management of bandwidth. A bonding group represents a collection of channels that form a single virtual channel. One or more service flows may be assigned to a bonding group. These service flows can be the same or different service categories. A resource allocation strategy must deal with all service categories in an appropriate manner, providing preference to those categories that require it. Fairness is typically only of concern between flows of similar service categories.

This work was supported in part by Cisco Systems, Inc. via Cisco Research Award *DOCSIS 3.0 Scheduling Algorithms*. The last author is supported in part by NSF CAREER award CCF-0845593.

Not surprisingly channel bonding poses significant implementation challenges. The scheduling algorithm must operate in two dimensions: time and frequency. In the downstream direction, the scheduler can stripe packets associated with a service flow over multiple channels in units that are either partial IP packets, exactly one packet, or multiple IP packets. Individual channels can be shared between potentially overlapping bonding groups of different sizes. To increase channel utilization, the Cable Modem Termination System (CMTS) can reassign service flows to different bonding groups. Further, a given cable modem (CM) can be switched to a different set of channels, but those channels must be single channels or an established bonding group. The standards do not specify the scheduling discipline to be used. While there has been a tremendous amount of prior research in the area of resource allocation and packet scheduling both in wired and wireless networks, the specific requirements of a DOCSIS 3.0 environment are unique and unstudied. Our broad research objective is to develop scheduling algorithms for a DOCSIS 3.0 environment. The first step must be to define the resource allocation strategy, which inherently requires a precise definition of fairness.

In this paper we present the necessary framework to study the problem. The objective of this research is to provide a feasible definition of fairness in a DOCSIS 3.0 channel bonding environment. The work reported in this paper provides the groundwork for further efforts in developing those resource allocation methods.

II. BACKGROUND

A. *DOCSIS 3.0*

The currently installed implementations of DOCSIS allow a CM to receive on a single channel and transmit on another single channel. This imposes a limit to the bandwidth available to an individual CM user. The newest version of this standard, DOCSIS 3.0, adds channel bonding capability such that a single CM can access multiple channels for transmission and reception of data. Some channels in a set are treated individually while other channels are assigned to bonding groups and treated as a single logical channel. Both upstream and downstream channels available to a CM can be organized as: 1) a single channel; 2) divided into bonding groups; 3) a

mix of bonding groups and individual channels. DOCSIS 3.0 allows a CM to receive on a single channel or bonding group and transmit on another single channel or bonding group. The DOCSIS 3.0 standard [1] allows a bit in the "Provisioned Attribute Mask" to indicate if a channel is treated as a single channel or is part of a bonding group. Without loss of functionality, in this study we treat individual channels as single channel bonding groups and treat the system as allowing a bonding group to consist of one or more channels.

DOCSIS provides different service levels to accommodate quality of service scheduling. This has primarily been accomplished by using different upstream access methods. For this reason, prior to DOCIS 3.0, most DOCSIS related scheduling research dealt with upstream scheduling. With the inclusion of channel bonding the downstream scheduling problem is no longer trivial. This study concentrates on the downstream scheduling problem as a first step. Fairness in scheduling only applies to flows within the same service category since the establishment of different service levels implies the need for some flows to be given priority without regard to fairness. However, the desire to maximize bandwidth utilization across multiple service categories requires the different levels of service to be taken into account.

In addition to higher bandwidth, channel bonding facilitates two useful functions: load balancing and service management. In a DOCSIS system all control of scheduling is centralized in the CMTS. As an example, a CMTS with six downstream channels that is managing CMs with four downstream channels each can control which four of the six channels each CM is using, within the established bonding groups, and adjust to maximize channel utilization. Different bonding groups can have attributes that can be used to easily differentiate service flows. For example, there might be a bonding group that is to be used only to broadcast video content, or a bonding group might be designated to be used by all subscriber peer-to-peer traffic. Maintaining fairness in this case is particularly challenging. Scheduling decisions made for a single bonding group can impact other bonding groups if the bonding groups intersect.

B. System Model

The channel bonding system input can be modeled as a set of three vectors describing demands, channel capacities, and the mapping between flows and channels. A demand vector D_i, $i = 1 \ldots n$, holds the individual bandwidth requests of n flows. A channel vector C_j, $j = 1 \ldots m$, holds the bandwidth of each of m channels. A two-dimensional binary channel map M_{ij} indicates the channels available to each flow; if $M_{ij} = 1$, then flow i is connected to channel j.

The output of the process is described by two vectors describing allocation of bandwidth to each flow and allocation of bandwidth to each channel. The vector A_i, $i = 1 \ldots n$, describes the assignments, where A_i is the allocation to flow i. Each entry CA_{ij} in a two-dimensional channel allocation map indicates the amount of flow i assigned to channel j.

C. Fairness

The early literature focused on the 'flow control' problem in packet switched networks [2] [3] [4] [5] [6]. Fairness was considered an outcome of a given flow control method. Gerla and Kleinrock [5] indicate that fairness is the fair allocation of resources among competing users and that unfairness is a natural byproduct of uncontrolled competition.

With a single channel of capacity C and n users the fair share allocation is simply C/n. Fair, in this case, means equal. However, there are flows that will require less than C/n. In this case the excess not required by one flow should be shared equally with all other flows. Keshav [7] defines an algorithm for this case of equal fairness:

- Resources are allocated in order of increasing demand.
- No source gets a resource share larger than its demand
- Sources with unsatisfied demands get an equal share of the resource.

After sorting the requests into increasing order, the total capacity of the channel is divided by the number of requests producing the equal fair share amount. The first, smallest, request is granted the lesser of this fair amount, or its request. The amount assigned is deducted from the total capacity amount. The process is then repeated until all requests have been assigned.

In a channel bonded system different users no longer have access to the same set of resources (a single channel). A given request could now be satisfied on more than one channel and that channel may, or may not, be shared with flows in another bonding group. These complications make it more difficult to determine what actually constitutes fairly sharing bandwidth. The initial problem, with a single channel, was only how much to allocate to each user. Now we must determine, for each user, not only how much, but how much on what channels. In this case providing equal sharing often means that available bandwidth will not be used even though it is accessible to some flows that have requested more than they received.

Jaffe [6] extends fairness to include different users operating over links of different capacities by claiming that a given allocation of bandwidth is fair if 1) each user's throughput is at least as large as all other users that share its bottleneck link, and 2) the only factor that prevents a user from obtaining higher throughput is the bottleneck link. This definition falls under the umbrella of the widely accepted 'max-min' approach to managing resources which require that flows get the same share of a bottleneck. This amounts to applying Keshav's algorithm to the flows using each bottleneck link, in turn, rather than applying it system wide. Max-min allocation gives preference to low bandwidth consuming flows by giving the maximum possible bandwidth to the source receiving the least among competing flows at a bottleneck.

Max-min allocation is a commonly used criterion for identifying the correct share of bandwidth allocated to flows in a network. Within the networking community this idea was originally (and independently) proposed by Jaffe [6] and Hayden [8]. The max-min criterion dictates that the smallest session must be as large as possible, and subsequently, the

second smallest session must be as large as possible, continuing until further allocations are not possible.

Max-min fairness was defined earlier in the algorithms community by Megiddo [9]. Although the term max-min fairness was not specifically used the algorithm was the same. It was shown that if all possible allocation vectors were sorted in increasing order the lexicographically greatest vector represents the max-min fair allocation. It was also shown that a max-min fair allocation will be a maximum flow for the network proving that achieving fairness does not require sacrificing throughput (in general, this property holds as long as we are dealing with a flow problem – such as the one under consideration here – that can be formulated with a single source or sink, rather than with multiple source-sink pairs).

In spite of the tremendous amount of related research, fairness is still a difficult concept to accurately define in actual networks. The complexity arises in part because of the many dimensions that must be considered. Fairness might be considered over a set of flows, individual user, or groups of users. Fairness criterion is likely to change over time. For example, fairness at packet transmission time scales is likely to be quite different from fairness expected over time scales of hours or days. Fairness is also inherently tied to the set of services that are delivered to users. Finally, defining fairness over sets of dissimilar yet competing flows is difficult. For example, how can one quantify fairness experienced by a VoIP flow versus flows associated with other applications such as web browsing or peer-to-peer?

D. *The resource allocation problem*

The DOCSIS 3.0 resource allocation problem begins with a high level view that deals with a set of flows of different service categories that need to be allocated to a set of channels that is broken up into sub-sets of channels called bonding groups. The DOCSIS standard defines service categories that provide, at the top end of the spectrum, a fixed TDM type service, down to a best effort service at the lower end of the spectrum. The resource allocation scheme must adhere to these service categories, providing preferential service to some flows over others as mandated by the categories themselves. At a lower level view, the resource allocation scheme must provide for flows of the same service category some manner of fairness between those flows. Because flows of both different and similar service categories can share either an entire bonding group, or some channels in a bonding group (or channels in several intersecting bonding groups), this presents a set of overlapping, and sometimes contradictory, goals.

This effort, which is a first step in this investigation, is to define this lower level view of the resource allocation problem where flows, all of the same service category, are being shared within a set of channels and potentially overlapping bonding groups. We deal with the simplified case of downstream flows, which eliminates the DOCSIS request grant mechanism needed for upstream flows and provides a situation where the CMTS has all information necessary to make scheduling decisions.

III. ALGORITHM DESCRIPTION

We conjectured that max-min fairness is an appropriate model to use for the initial DOCSIS 3.0 scenario considered here, since it provides both a fair allocation of bandwidth as well as maximum utilization of the available bandwidth. As part of this preliminary work, a flow network model of the channel bonding system was built to implement Megiddo's max-min fair algorithm and calculate the fair flow allocations. This was implemented in two phases. The first was to find an allocation for which min $\{A_i : i = 1 \ldots n\}$ is as large as possible. That is, we wish to find an allocation maximizing the amount of bandwidth received by all flows. The second phase then finds a max-min fair allocation (in which the allocation vector A, sorted in increasing order, is lexicographically maximal), by repeated application of the preceding algorithm. This max-min fair allocation will also maximize the total bandwidth allocated to all flows.

A. *Flow Network Model*

A flow network [10] [11] G = (V, E) is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity c_{uv}. There is a single source vertex *s* and a single sink vertex *t*. The maximum flow problem involves determining the maximum flow through the network from *s* to *t*, subject to the capacity constraints of all edges. The Ford-Fulkerson algorithm [12] can be used to find the maximum flow through the network. The Ford-Fulkerson algorithm also provides the flow allocation across each edge (or in our case channel) of the graph in the maximum flow.

A flow network graph can be constructed to model the channel bonded system by starting with a bipartite graph with a left vertex for each flow and a right vertex for each channel. Edges are drawn from each individual flow vertex to the channels that the flow has access to. The single source vertex *s* has an edge to each flow vertex, and each channel vertex has an edge to the single sink vertex *t*. Fig. 1 shows an example network where flow 1 can access channels 1 and 2, flow 2 can access channel 2, and flow n can access channels 2 and m.

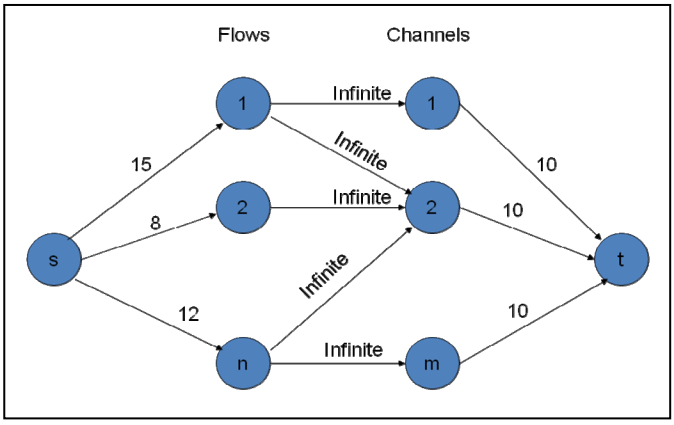

Figure 1. Flow Network for channel bonded system

The capacities for each edge are shown on the graph. Each source to flow edge is given a capacity equal to the bandwidth request amount of the corresponding flow, the values in the demand vector D. In this example D = (15, 8, 12). Each channel to sink edge is given a capacity equal to the bandwidth of that particular channel, the values in the channel vector C. Again, for this example C = (10, 10, 10). Each flow to channel edge is given infinite capacity, so that the flow is dependent only on the request amounts and the channel bandwidth. Running the Ford-Fulkerson algorithm on the resulting network will yield the maximum flow through this network.

In this example, using a breadth first search to find the augmenting paths in the Ford-Fulkerson algorithm, the allocation vector is A = (15, 5, 10) and the maximum flow value is 30. This can be verified by the max-flow min-cut theorem, which states that the maximum flow through a network is equal to the aggregate capacity of a minimum cut separating the source from the sink. In our example, a minimum cut of capacity 30 (matching the value of our flow) separates all vertices but the sink from the sink.

B. Finding Fair Allocations

As was the case above, finding a maximum flow through this network will not usually provide a fair share assignment. The maximum flow problem isn't concerned with fairness, only aggregate throughput. However Megiddo's approach shows how to determine a fair allocation by solving a succession of maximum flow problems.

The first step in Megiddo's max-min fair allocation algorithm is to find an allocation A whose minimum component is as large as possible. That is, we wish to maximize the value of x such that we can find an allocation vector with $A_i \geq x$ for i = 1…n. Perhaps the simplest approach for this problem is to binary search on x. In each step of the binary search, we set x to the capacity of each outgoing edge from the source, and we then check if a maximum flow fully saturates all of these edges. If not, our guess for x was too high, since there is no way to allocate at least x units to every flow, so we revise our guess for x downwards. Otherwise, our guess was correct or too low. In theory, this approach might loop forever if the final maximum value of x is a number like 1/3 that has no exact binary representation. However, in practice, we can terminate the binary search once it has determined x to within some desired tolerance. One can also apply more sophisticated algorithms to determine x more efficiently; for example, Gallo, Grigoriadis and Tarjan[13] describe a more complicated algorithm for solving this problem (which they call a *parametric* maximum flow problem) in the same worst-case running time as a single standard maximum flow problem. In either approach, we limit the initial range for x to [0, min D_i], since a higher value would unnecessarily allocate more bandwidth to some flow i than its demand D_i.

Once we have determined the maximum value of x (possibly equal to the minimum demand) such that all flows can be allocated at least x units of bandwidth, we will have reached a bottleneck point where further increases in x need to "leave some flows behind", as there is no way to allocate more than x units of bandwidth across the board to every flow. Specifically, there will be some subset $S \subseteq \{1, \ldots, n\}$ of flows for which $x = D_i$ for all $i \in S$, or if this is not the case, then there must exist some subset S of flows such that it is impossible to allocate strictly more than x units of bandwidth to all flows $i \in S$ simultaneously. In the second case, we can locate S using the maximum flow minimum cut theorem, by including flow i in S if and only if all of the channels available for use by flow i are completely saturated by our maximum flow solution (since we might have terminated our binary search on x early to avoid an infinite loop, we must treat nearly saturated channels as saturated for this purpose). Note that the flows in S are those that cannot be unilaterally increased, while the remaining flows may still be able to accept higher bandwidth allocations. We therefore "freeze" the flows in S, never again raising their associated capacities in our flow network. For the remaining flows, we repeat the entire process again, increasing their allocations by binary searching for a new value of x such that all flows (excluding those frozen in S) can be assigned at least x units of bandwidth. We then identify a second set of "frozen" flows, and repeat the process iteratively until all flows are finally assigned. The final result will give an approximate max-min fair allocation (approximate due to the fact that we may terminate our binary searches early) that is also a maximum flow.

Since all flows do not have access to the same set of resources (channels), it is possible to find the max-min fair amount that can be provided to every flow and still have bandwidth remaining on some channels. This can occur when demand from all flows connected to a given channel is less than the capacity of that channel. For example, if the demand vector in Fig. 1 were changed to D = (15, 8, 5), since channel 3 would only be connected to flow *n* with a demand only equal to half the channel capacity, the remainder of channel *m* would be unusable. Without changing the bonding group assignments this bandwidth cannot be utilized. This rebalancing of the loads by dynamic reassignment of bonding groups is one of our future planned research areas.

C. Results

The flow network graph was coded to implement the max-min fair algorithm. It was then tested to provide allocations for each flow to channels to achieve max-min throughput. Fig. 2 is an example result showing the output of the algorithm. In this example there are ten flows all with demand 1000, and four channels all with capacity 1000. The map M provides a channel assignment designed to prove the lexicographically maximum assignment of the allocations. This example has three single channel bonding groups (1, 2 and 3) which all intersect multi-channel bonding groups. In addition there are three multi-channel bonding groups (channels 1 and 2, channels 2 and 3, and channels 3 and 4). The bottleneck link (channel) with the largest number of flows is channel 4.

D = (1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000)

C = (1000,1000,1000,1000)

	Max-min fair allocations			
	C_1	C_2	C_3	C_4
M = ((1,1,0,0)	1000			
(0,1,0,0)		500		
(0,1,1,0)		500		
(0,0,1,0)			333	
(0,0,1,0)			333	
(0,0,1,1)			333	
(0,0,0,1)				250
(0,0,0,1)				250
(0,0,0,1)				250
(0,0,0,1))				250

Figure 2. Allocation Example

It can be seen that the initial fair allocation was 250 to all ten flows. At that point the bottleneck at channel 4 is settled. The algorithm continues with the flows that have access to channels 1, 2, and 3. Next the channel 3 bottleneck is divided with 333 to each flow. The process then continues with the flows having access to channels 1 and 2. Those two flows get 500 each, leaving capacity only on channel 1. The increase continues until channel 1 is full at 1000.

IV. Conclusions And Future Work

A flow network graph was implemented to model a DOCSIS 3.0 network and the max-min fair algorithm was used to determine fair allocations for each flow. Additionally, the algorithm provided channel assignments for each flow. This effort provides a baseline from which to continue the scheduling effort for DOCSIS 3.0. This process provided bit-by-bit assignments, however the actual system operates on a packet-by-packet basis. The results of this effort will be used in evaluating how accurately various packet-by-packet scheduling disciplines perform relative to this ideal case.

Future efforts will follow on two fronts. The first is the implementation of actual scheduling disciplines. Preliminary work has started using forms of deficit round robin and implementations of weighted fair queuing to evaluate how well they perform against this baseline.

The second area of concentration will be in evaluating algorithms to optimize the use of bonding groups. In the case where bandwidth can not be utilized because it is not accessible the only solution is to rebalance the loading of the bonding groups. We will evaluate this possibility on two levels. The first is the dynamic reassignment of channels to different bonding groups. Currently the bonding group mappings are manually constructed. The goal is to attempt to provide assignments which will evenly distribute the load across the bonding groups based on dynamic operational throughputs. The present algorithm provides the input to trigger the system that the bonding groups should be rebalanced.

Due to the time needed to switch a CM from one channel to another, a second effort will look at alternative ways in which to optimize the initial setup of bonding groups such that the need to switch between channels can be minimized.

Acknowledgment

We acknowledge our sponsors at Cisco, in particular Alon Bernstein, in providing insight into the DOCSIS 3.0 scheduling problem.

References

[1] http://www.cablelabs.com/specifications/CM-SP-MULPIv3.0-I11-091002.pdf

[2] D. Davies, “The Control of Congestion in Packet-Switching Networks,” IEEE Transactions on Communications, vol. COM-20, June 1972.

[3] J. Gray, “Network Services in Systems Network Architecture,” IEEE Transactions on Communications, vol. COM-25, pp. 104-116, 1977.

[4] V. Ahuja, “Routing and Flow Control in Systems Network Architecture,” IBM Systems Journal, vol. 18, no. 2, pp. 298-314, 1979.

[5] M. Gerla and L. Kleinrock, “Flow Control: A Comparative Survey,” IEEE/ACM Transactions on Communications, 28(4), pp. 553–574, 1980.

[6] J. Jaffe, “Bottleneck Flow Control,” IEEE Transactions on Communications, 29(7), pp. 954-962, 1981.

[7] S. Keshav, An Engineering Approach to Computer Networking. Addison Wesley, 1997

[8] H. Hayden, Voice Flow Control in integrated Packet Networks, Master’s Thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering, Cambridge, MA, 1981.

[9] N. Megiddo, “Optimal Flows in Networks with multiple Sources and Sinks,” Mathematical Programming, 7:97-107, 1974.

[10] T. Cormen, C. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms, McGraw-Hill, second edition, 2001.

[11] R. Ahuja, T. Magnanti and J. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, 1993.

[12] L. Ford and D. Fulkerson, “Maximal Flow through a Network”, Canadian Journal of Mathematics, 1956

[13] G. Gallo, M. Grigoriadis and R. Tarjan, “A fast parametric maximum flow algorithm and applications,” SIAM Journal of Computing, vol. 18, no. 1, pp. 30-55, 1989.