Verifying Home Network Bandwidth Sharing Plans

Dimosthenis Pediaditakis
Department of Computing
Imperial College London
London, SW7 2AZ, United Kindgdom
Email: d.pediaditakis@imperial.ac.uk

Naranker Dulay
Department of Computing
Imperial College London
London, SW7 2AZ, United Kindgdom
Email: n.dulay@imperial.ac.uk

***Abstract*—Experimental evidence from recent measurement studies has shown that bandwidth bottlenecks usually reside at the edges of the Internet, which is also true for residential networks where users share network resources and there is a need to regulate the usage of bandwidth. In this paper we introduce a rule-based approach for specifying bandwidth sharing plans for home networks which are enforced in a distributed manner across the network. More specifically we focus on the problem of verifying these sharing plans detecting potential inconsistencies which may arise from the rules that are specified by users. We describe a novel tree-based structure to model and verify the network's sharing scheme and support the specification of custom conflict resolution policies.**

I. Introduction

Despite faster broadband technologies, there are a number of reasons why home users still encounter situations where bandwidth is inadequate to meet their needs. First, service requirements continue to grow, for example, media streaming, online gaming, software downloads and P2P file sharing. Second, increasing numbers of Internet-enabled devices are being introduced into homes. Third, one or two "heavy" users' often account for most of the total data volume [1]. Fourth, the advertised speeds of broadband connections rarely reflect a connection's quality ([2], [3]). Fifth, the traffic of home networks is often bursty and more intense than in enterprise environments [4].

Typically, home network gateways forward packets using best-effort schemes and aim to achieve fairness on a per packet basis. Quality of service guarantees are only provided on a very coarse-grained basis, using the "type of service" (TOS) field of IP datagrams. Based on the value of the TOS field, traffic is mapped to predefined priority classes, each reflecting a different forwarding behaviour. Such functionality is limited in the sense that classes are predefined and protocols are mapped by default to a particular class. They do not support the explicit allocation of bandwidth resources, and in practice home users do not understand and are unable to configure this complicated mechanism.

We believe that there is an unstated requirement for home networks to support differentiated services on a per user, per service or per application basis.

In this paper, we consider the problem of statically allocating bandwidth resources of home networks. We do not deal with queuing models and adaptive bandwidth control algorithms that actively control the delays and the rates among flows. Our intention is to apply traffic shaping in such a way that it reflects the requirements and priorities among users. We can cap, assure or explicitly assign to hosts and services any given amount of bandwidth, through a simple rule language. However, since users can create arbitrary rules, we need to ensure that each rule is correctly specified and that rules do not interfere with each other, both at the network level and the host level.

The paper makes the following contributions. First, we introduce the concept of a bandwidth sharing plan consisting of a number of bandwidth brokering rules which are supplied by home users. Second, we identify and define the key properties that a bandwidth sharing plan should have in order to be "sound". Third, we introduce a novel tree-based structure to model and verify that a bandwidth sharing plan is sound.

The rest of this paper is organized as follows; in section II we describe in detail the problem of verifying bandwidth sharing plans. In section III we formally define the desired properties that a sharing plan should have and enumerate the potential conflicts that may arise. Section IV outlines our tree-based model used to represent plans and provide the details of our verification procedure. The overall architecture of our system and implementation details are presented in section V along with preliminary measurements. The last two sections discuss related work and conclude.

II. Problem Description

A. Assumptions

Home networks demonstrate unique connectivity characteristics [5] with substantial wireless link variations over time. Similarly, broadband links may experience capacity variations over time, due to a multitude of reasons, including external sources of interference, cabling issues, high attenuation, loss of synchronization and many more. It turns out that the bandwidth bottlenecks tend to appear near or even inside homes and they may "move" a few hops forward or backwards over time ([6], [3], [7]). In this paper, we assume that the available bandwidth (when the network is idle) is fixed and is the same for all users, regardless of the type of connection they have to the default gateway. We also assume that any bottleneck is common for all users and is located at the WAN link.

B. Bandwidth brokering rules

Bandwidth brokering rules express resource allocation tasks and consist of three distinct clauses: *Task*, *Subject* and *Service*. As an example, the rule "Cap the downlink rate of local host 10.0.1.1 at 3 Mpbs, but only for HTTP sessions with remote host www.downloads.com", is equivalent to the bandwidth brokering rule:

Task $\{Cap,\ Download,\ 3,\ Mbps\}$ **Subject** $\{IP,\ 10.0.1.1\}$
Service $\{host,\ www.downloads.com\ TCP,\ s:80\}$

There are a variety of shortcuts and our tools can make intelligent guesses for missing values, e.g. TCP s:80 would be inferred from www.downloads.com.

Currently, we support three types of bandwidth specification to reflect the different requirements:

- **Maximum rate** ($\lceil R_i \rceil$): This is the maximum amount of bandwidth that a local host may use under any circumstance. It is actually a cap-limit that prevents some users from using the full speed of the network.
- **Minimum assured rate** ($\lfloor R_i \rfloor$): This is the minimum amount of bandwidth that will be available anytime for a local host. If the network resources are not used by the host, other devices may "borrow" them. This particular type of allocation rule, may sometimes introduce small initial delays while waiting for potentially full queues to serve again some empty slots.
- **Exclusively assigned rate** ($\|R_i\|$): This statically assigns the specified rate to a local host. Unlike the minimum assured rate, other hosts cannot "borrow" the unused resources, and thus delays are considerably smaller providing better QoS support.

Assured and exclusive rate allocations are treated as two separate co-existing bandwidth channels, and are mutually exclusive. Both types of allocation cannot be applied on the same managed class. However, an exclusive rate allocation for a more specific class (e.g. protocol p for Host X) will override and cancel assured rates which are possibly allocated to more inclusive classes (e.g. for Host X in general). This holds vice-versa also.

C. Verifying plans

From a usability point of view, providing users with the ability to manage a home network's resources is a desired feature. In a typical scenario users may compose, add, activate, disable and remove bandwidth brokering rules via an intuitive user interface in order to fulfill particular needs. This however can lead to situations where the specified rules might contain invalid parameters, contradict each other or overwhelm the network's capabilities. For example, a rule which limits the downlink rate for a host to 3 Mbps, and at the same time another rule which provides the broadband connection an exclusive rate of 5 Mbps might result in wasting 2 Mbps. It is thus important to have a mechanism which can help avoid such inconsistencies.

In this paper, we only consider the problem of assuring that the addition of a rule to the global bandwidth sharing plan, does not conflict with existing rules. Also, an early check is performed during composition, making sure that the supplied values fit the characteristics of a given home network.

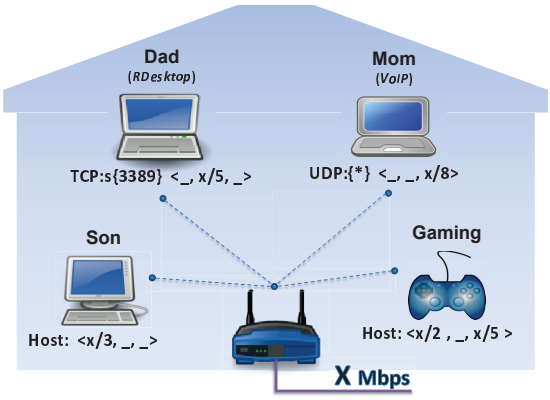


Fig. 1. A bandwidth sharing scenario

D. Example scenario

Throughout the rest of the paper, we will use the scenario illustrated in Fig. 1, consisting of a number of devices used by users with different requirements. User Dad is mostly doing office work using a remote desktop application, user Mom is mainly using her netbook for conversations over VoIP, Son makes heavy use of P2P file sharing applications, and finally the gaming console is used during the weekends for playing multiplayer internet games. Depending on the usage profile, the QoS requirements for each user are different. Some services are more sensitive to delays (e.g. VoIP, multiplayer gaming), while others may require certain amounts of bandwidth for a good quality of experience. The tuples shown under each device, represent the bandwidth brokering rules for each user and are provided in a condensed form. The first value represents the maximum allowed (ceiling) rate, the second represents the minimum assured rate and the third stands for the exclusively assigned rate. Applications with minimum delay and bandwidth requirements are assigned exclusive rates, while others that are more delay tolerant are assigned minimum assured rates. X is the total available downlink bandwidth at the main gateway.

III. Desired sharing properties

A bandwidth sharing plan consists of rules that assign rates among hosts, subnets and services (e.g. cap $Host\ A$ to Y Mbps) but they may also specify bandwidth management tasks that further split the shares among services for a given network entity. This is a hierarchical split of network resources among several network entities, each representing a particular network or protocol. Our aim is to make sure that this "split", which is based on user-defined brokering rules, does not contain inconsistencies. In this section we define the "good" sharing properties that a plan should preserve, so that the overall sharing plans reflect the network behaviour expected by users.

We consider home network environments where the bottleneck is the WAN link at anytime and the local communication

TABLE I
BANDWIDTH SHARING PLAN TERMINOLOGY

Term	Meaning
R_{GW}	Total available bandwidth at the gateway
R_{def}	Available bandwidth for the default traffic
a_{def}	Minimum R_{GW} fraction for the default traffic
R_{MIN_def}	Minimum bandwidth for the default traffic
Subnets	
N	The set of all subnets in the home network
$\lceil R_{Nj} \rceil$	Maximum allowed (ceiling) rate for subnet Nj
$\lfloor R_{Nj} \rfloor$	Minimum assured rate for subnet Nj
$\parallel R_{Nj} \parallel$	Exclusively assigned rate for subnet Nj
Hosts	
H	The set of all hosts in the home network
$\lceil R_{Hi} \rceil$	Maximum allowed (ceiling) rate for host Hi
$\lfloor R_{Hi} \rfloor$	Minimum assured rate for host Hi
$\parallel R_{Hi} \parallel$	Exclusively assigned rate for host Hi
Services	
S	The set of all services
$\lceil R_{Sk} \rceil$	Maximum allowed (ceiling) rate for service k
$\lfloor R_{Sk} \rfloor$	Minimum assured rate for service k
$\parallel R_{Sk} \parallel$	Exclusively assigned rate for service k

links have a high capacity which is fixed over time. The following properties refer to the bandwidth resources of a home's main gateway downlink channel, and similarly, they can be applied on the Internet uplink channel and on local communications.

A. Home network wide properties

The terms we use throughout this section are given in TABLE I. Note that all terms refer to the aggregate amount of all the bandwidth allocations that have resulted from the bandwidth sharing rules introduced by users. They do not reflect the amount of resources specified by each rule separately.

Property 1: The "default traffic", which is not subject to any bandwidth brokering rule, should always be assigned a non-zero rate (R_{MIN_def}). This rate is defined as a fraction (a_{def}) of the total available bandwidth at the network's gateway (R_{GW}):

$$R_{MIN\_def} = a_{def} * R_{GW}, \;\; 0 < a_{def} \leq 1 \tag{1}$$

$$R_{MIN\_def} \leq R_{def} \leq R_{GW} \tag{2}$$

Property 2: The total amount of the minimum assured rate ($\lfloor R_{Nj} \rfloor$) and exclusively assigned rate ($\parallel R_{Nj} \parallel$) for all subnets, along with the available bandwidth for the "default traffic", should add up to the total available bandwidth at the network's gateway:

$$\sum_{j=1}^{m} (\lfloor R_{Nj} \rfloor + \parallel R_{Nj} \parallel) \; + R_{def} = \; R_{GW}, \;\; \forall N_j \in N \tag{3}$$

From properties 1 and 2, the total amount of the minimum assured and exclusively assigned rates for all subnets is always less than the total available bandwidth at the network's gateway:

$$\sum_{j=1}^{m} (\lfloor R_{Nj} \rfloor + \parallel R_{Nj} \parallel) \; < \; R_{GW}, \;\; \forall N_j \in N \tag{4}$$

B. Subnet-level properties

Property 3: Any subnet ceiling rate ($\lceil R_{Nj} \rceil$) should be no greater than the total available bandwidth at the network's gateway:

$$\lceil R_{Nj} \rceil \; \leq \; R_{GW}, \; \forall N_j \in N \tag{5}$$

From equation 4 and property 3, the ceiling rate, the minimum assured rate and the exclusively assigned rate for any subnet $\lceil R_{Nj} \rceil$, should be no greater than the gateway's total available bandwidth:

$$\lceil R_{Nj} \rceil, \lfloor R_{Nj} \rfloor, \parallel R_{Nj} \parallel \leq R_{GW}, \; \forall N_j \in N \tag{6}$$

Property 4: Assured and exclusively assigned bandwidth resources for any subnet, should be utilized at their maximum rates. In other words, for any subnet it should hold that the sum of the minimum assured and the exclusively assigned rates is no greater than their respective ceiling rate:

$$\lfloor R_{Nj} \rfloor + \; \parallel R_{Nj} \parallel \leq \; \lceil R_{Nj} \rceil, \; \forall N_j \in N \tag{7}$$

Property 5: The sum of all the assured rates for hosts in a subnet, should not exceed the subnet's overall assured rate:

$$\sum_{i=1}^{n} \lfloor R_{Hi} \rfloor \leq \; \lfloor R_{Nj} \rfloor, \;\; \forall H_j \in N_j, \; N_j \in N \tag{8}$$

Property 6: The sum of all the exclusively assigned rates to hosts in a subnet, should not exceed the subnet's overall exclusive rate:

$$\sum_{i=1}^{n} \parallel R_{Hi} \parallel \leq \parallel R_{Nj} \parallel, \;\; \forall H_i \in N_j, \; N_j \in N \tag{9}$$

Property 7: The maximum cap rate of any host in a given subnet, should be less than or equal to a subnet's cap rate:

$$0 \; \leq \; \lceil R_{Hi} \rceil \; \leq \; \lceil R_{Nj} \rceil, \;\; \forall H_i \in N_j, \; N_j \in N \tag{10}$$

From equation 6 and properties 2, 5 and 6, we get the bounds for all host level assignments:

$$\sum_{i=1}^{n} (\lfloor R_{Hi} \rfloor + \parallel R_{Hi} \parallel) \; \leq \; R_{GW} - R_{def}, \; \forall H_i \in H \tag{11}$$

C. Host-level properties

Property 8: The equivalent of property 4, should also hold for each host separately. For any host, the sum of the minimum assured and the exclusively assigned rates should be no greater than its respective ceiling rate:

$$\lfloor R_{Hi} \rfloor + \; \parallel R_{Hi} \parallel \leq \; \lceil R_{Hi} \rceil, \; \forall H_i \in H \tag{12}$$

Property 9: The sum of all the assured rates for services of a particular host should not exceed the host's overall assured rate:

$$\sum_{k=1}^{t} \lfloor R_{Sk} \rfloor \leq \lfloor R_{Hi} \rfloor, \ \forall S_k \in S, \ H_i \in H \qquad (13)$$

Property 10: The sum of all the exclusively assigned rates for services of a particular host should not exceed the host's overall exclusive rate:

$$\sum_{k=1}^{t} \parallel R_{Sk} \parallel \leq \parallel R_{Hi} \parallel, \ \forall S_k \in S, \ H_i \in H \qquad (14)$$

Property 11: The maximum cap rate of any service for a given host should be less than or equal to the host's cap rate:

$$0 \leq \lceil R_{Sk} \rceil \leq \lceil R_{Hi} \rceil, \ \forall S_k \in S, \ H_i \in H \qquad (15)$$

Property 12: Following the same line of property 8, for any given host the sum of the minimum assured and the exclusively assigned rates to a service should not exceed its respective ceiling rate:

$$\lfloor R_{Sk} \rfloor + \parallel R_{Sk} \parallel \leq \lceil R_{Sk} \rceil, \ \forall S_k \in S, \ \ for\ any\ host \qquad (16)$$

IV. Verification Procedure

Considering the semantics of a particular rule, we can have clear expectations from the network behaviour, with respect to a given user or service. At the network level however, things get complicated. Users would expect the overall network behaviour to comply with each of the contracts of the sharing plan, under all circumstances. However, this is only possible if allocations do not overwhelm the available network resources at anytime and if the sharing plan is free of conflicts. In order to reason about the network-level behaviour, it is important to preserve the desired sharing properties in section III so that we have a network with a "clean" (conflict-free) sharing model.

Whenever a user requests to activate a bandwidth brokering rule, a verification procedure is triggered to sanity check the rule and check for conflicts with the existing rule-set. If verification is successful, the rule's definition is handed to a distributed enforcement service, otherwise, the actions taken depend on the system's conflict-resolution policy.

A. The tree based model

We employ a tree-based structure to model the home network's bandwidth sharing plan (Fig. 2). Normally there are three distinct trees, one for Internet uploads, one for Internet downloads, and one for LAN communications. In this paper we only consider the tree for Internet downloads but similar techniques apply to the other trees. Network traffic which is subject to any type of contract (maximum, assured or exclusive rate) is referred to as managed; the remaining traffic is referred to as default or unmanaged. Each tree level maps to a different network view. Nodes nearer the root are more generic; they get more specific as we move away from the root. We use two types of nodes, the structure nodes which are intermediate nodes (rectangular) and the outer-leaf nodes (circular).

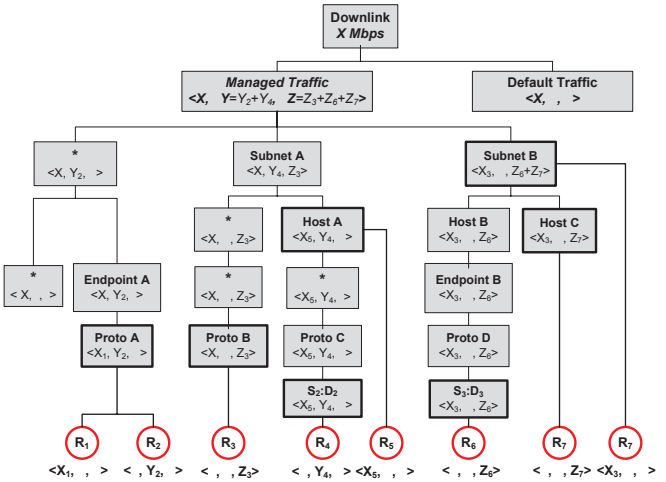


Fig. 2. The tree model for download

Structure nodes represent a logical mapping between bandwidth shares and network elements. Structure nodes at depths from 0 (root) to 6 represent the available bandwidth for downloads, the overall managed traffic, source subnets, source hosts, destination hosts (Internet endpoints), protocols and source-destination port numbers respectively. Structure nodes at depth 1 and higher, are aggregates of the bandwidth sharing rates. These nodes, do not represent actual rules, but keep track of the bandwidth allocations of the sub-trees attached to them, and also the cap limits of their ancestors. They maintain a triplet of values for the maximum, aggregate assured and aggregate exclusive rates. These values are updated whenever a new rule is attached to the tree. Structure nodes are created dynamically, the time a rule is successfully activated by the user. In Fig. 2 you can observe that some special nodes are labeled with "*"; these represent all possible instances of elements at the current depth for the given subtree they belong. For example, the node labeled with "*" on the left of "Host A", represents all the nodes in "Subnet A", and the node below that represents all potential endpoints.

The second type of nodes are "outer nodes" which hold the actual bandwidth brokering rules. They are always leaf-nodes, attached under the appropriate structure node. Depending on the semantics of the rule definition, the appropriate path of ancestor structure nodes is created if it does not already exist. Attaching outer nodes on the tree usually affects the overall bandwidth sharing model, propagating changes up to the root if necessary.

B. Verification procedure

If a user requests to activate a particular rule (R) given a snapshot of the tree model (T), the Verify procedure (Fig. 3) is triggered and if successful, the rule is added as an outer node in the model. A distributed enforcement service then takes over, distributing and applying the rule's management action on the appropriate network device(s). Traversal and conflict checking procedures are explained in the following sections.

Algorithm 1 The *Verify* procedure

```
1:  procedure Verify (R, T)        {R: rule definition, T: tree model}
2:  begin
3:  tmp := traverse(T, R)
4:  switch tmp.type
5:   case Max:      cnf := checkMax(R, tmp, T)
6:   case Assure:   cnf := checkAssured(R, tmp, T)
7:   case Exclusive: cnf := checkExclusive(R, tmp, T)
8:  endSwitch                                  {cnf: returned conflicts}
9:  if cnf ≠ NIL then                          {no conflicts}
10:     discardChanges(T)
11:     resolveConflict(R, cnf, T)             {Resolve conflicts}
12: commitChanges(T)
13: end
```

Fig. 3. Top-level verification procedure

Algorithm 2 The *checkMax* procedure

```
1:  procedure checkMax (R, n, T)
    {R: rule definition, n: starting node, T: tree model}
2:  begin
3:  if R.max > root(T).rate then                {Check-1}
4:      return root(T)                          {Unreasonable cap}
5:  if R.max > n.max then                       {Check-2}
6:      return findRule(Max, n)
7:  if R.max < n.assured then                   {Check-3}
8:      return findRules(Assured, n)
9:  if R.max < n.exclusive then                 {Check-4}
10:     return findRules(Exclusive, n)
11: if R.max < n.assured + n.exclusive then
12:     foreach i in children(n) do
13:         tmp := checkMax(R, i, T)            {Recursion}
14:         if tmp ≠ NIL then
15:             return tmp                      {Subtree conflict}
16:     endForeach                              {No subtree conflict detected}
17:     return n                                {Flag joint conflict here}
18: n.tmp.max := R.max                          {Temporary change}
19: return NIL                                  {No conflicts found}
20: end
```

Fig. 4. The procedure that checks conflicts for maximum rates

C. Tree traversal

Given a rule definition (R) and a tree model instance (T), the traverse procedure creates an outer node based on R and finds the appropriate structure node to attach it. The tree is built dynamically. If a proper structure node for rule R does not exist, the traversal creates the missing nodes. All changes made to the model are temporary and get committed when the appropriate checks are passed. The procedure finally returns the temporary outer node which contains the rule specification and waits to be verified against the rest of the model.

After traversal finishes, the verification procedure (Fig. 3) checks whether the new rule violates any of the properties described in section III. This check is done recursively in both directions (towards the root and towards the leaves), starting from the node returned by "traverse". At each depth, the rule is checked against the aggregated information in structure nodes. If a disagreement is identified, a conflict is flagged, otherwise, the current structure node is updated appropriately, and the recursion continues upwards. If no conflicts are detected, the outer node is permanently attached to the tree and any changes made to the aggregates in structure nodes get "committed".

Algorithm 3 The *checkAssured* procedure

```
1:  procedure checkAssured (R, n, T)
2:  begin
3:  {R: rule definition, n: starting node, T: tree model}
4:  directRule := getRule(Assured || Exclusive, n)
5:  if directRule ≠ NIL then                    {Check-1}
6:      return directRule          {Any directly attached rule conflicts}
7:  cnf := checkSubtreeAssured(R, n)            {Check-2: subtree}
8:  if cnf ≠ NIL then
9:      return cnf                 {starvation conflict is detected}
10: if R.assured + n.assured ≥ T.root.rate then {Check-3}
11:     return T.root                {Overwhelming allocation}
12: extra := R.assured − n.assured
13: return AllocateAssured(extra, n, T)     {Check-4: availability}
14: end

15: procedure checkSubtreeAssured (R, n)
16: begin
17: if R.assured ≥ n.assured then
18:     return NIL                           {No starvation detected}
19: directRule := getRule(Assured, n)
20: if directRule ≠ NIL then          {Always NIL for subtree's root}
21:     if R.assured < directRule.assured then
22:         return directRule        {Starvation: directly attached rule}
23: foreach i in children(n) do
24:     tmp := checkSubtreeAssured(R, i)        {Recursion}
25:     if tmp ≠ NIL then
26:         return tmp
27: endForeach
28: return n                        {Combined starvation at node n}
29: end

30: procedure AllocateAssured (allocReq, n, T)
31: begin
32: if n = T.root then
33:     return NIL                           {Allocation is successful}
34: totalAlloc := allocReq + n.assured + n.exclusive
35: if totalAlloc ≥ n.max then
36:     return findRule(Max, n)      {Cap lower than allocations}
37: if parent(n) = T.root then       {Global Managed Traffic node}
38:     if totalAlloc > T.root.rate−MinBW_Default then
39:         return T.root            {Conflict: not enough bandwidth}
40: n.tmp.assured := n.assured + allocReq
41: return AllocateAssured(allocReq, parent(n), T)
42: end
```

Fig. 5. The routine that checks conflicts for assured rates

D. Checking maximum rate contracts

Depending on the type of bandwidth brokering rule, the checking procedure at a structure node differs. The procedure for checking maximum rate cap limits is listed in Fig. 4. First, any cap limit values over a gateway's available bandwidth are discarded (Check-1). If the current structure node "n" has a cap rate lower than the rule, then some existing rule along the path towards the root has introduced that limit (Check-2). Cap limits are inherited from a parent node to the whole subtree under it. Procedure "findRules" returns the outer node containing the rule that introduced the maximum rate. Checks 3 and 4 test whether properties 4, 8 and 12 (see section III) are violated. All the conflicting outer nodes are returned. Finally, Check-5 repeats the procedure recursively for the subtree under node "n". If no conflicts are detected, then the temporary changes in the node's maximum cap rate are made permanent (line 18).

In Fig. 6 we show the tree model after applying the rules

for the example scenario (section II-D). Rules "R_1" and "R_2" are of type maximum cap rate. Observe that their activation is successful because their specified rates do not exceed the values of Son and "Console" host nodes which were inherited directly from "Managed Traffic" via "Subnet A".

E. Checking assured rate contracts

Checking a request for the allocation of an assured rate is more complex than checking maximum rates. The procedure is shown in Fig.5, and consists of one main procedure which calls two recursive sub-procedures. Check-1 checks for directly attached outer nodes (bandwidth allocation rules) which by definition will conflict with the activated rule, since they refer to the same managed object. If the procedure continues with Check-2, then the current node's assured rate value, is due to some allocation in the subtree under node "n". Procedure "*checkSubtreeAssured*" locates the conflicting rule(s), returning as much specific information as possible. The conflict with the subtree may be because of a number of accumulated allocations (line 23). Assured rates in structure nodes are aggregates of subtree allocations and of directly attached rules (outer nodes). Continuing, line 10 is reached if no conflicts have been detected so far. At this point, it is important to make sure that there are adequate bandwidth resources to perform the new allocations. Checks 3 and 4 perform this test by propagating recursively the allocation request up to the root of the tree. The "*checkAssured*" procedure detects any potential violation of properties 2, 4, 5, 8, 9 and 12 (see section III).

An example of assured rate allocation, is $Dad's$ rule (R_2) (see Fig.6) for the remote desktop application. Clearly, there are plenty of resources available to cover the allocation of $X/6$ of total bandwidth. The rule is directly attached (as an outer node) under structure node "3389:*" and the assured rate is propagated up to the "Managed Traffic" node.

F. Checking exclusive rate contracts

The procedure for activating rules for exclusively assigned rates is omitted because it is almost identical with the one for assured rates. The reason is that the two types of allocation are treated as separate channels. However, a common check is performed for both (line 34 in Fig.5), to preserve properties 2, 4, 8 and 12. In our example scenario, $Mom's$ host requests an exclusive rate of $X/8$ for VoIP, because it requires minimum delay and a small amount of assured bandwidth. Again, the request can be performed (Fig.6), since there are enough resources both for R_2 and R_5 allocations.

G. Disabling a rule

Disabling a rule is not as straight-forward as removing the respective outer node from the tree model. Depending on the rule's type, different actions are performed. When disabling maximum rate rules, first the outer node is marked for removal, second the attached structure node's accounting for max rate is updated based on parent's value, and third, the second step is repeated recursively for each node of the subtree. On the other hand, when evicting assured and exclusive rate allocations, first

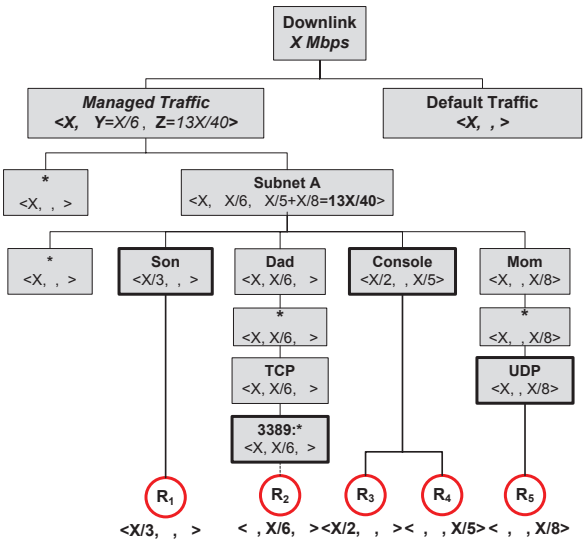


Fig. 6. The tree model for the example scenario

the rule's outer node is marked for removal, second, structure node's accounting for assigned rates are updated with just the sum of ancestors' values, and finally, the second step is performed recursively on each node upwards towards to the root. In both cases, any change in the model is marked in order to keep track of the rules "affected" from this process. This information is used by the enforcement service which may need to update the configuration on the actual network.

H. Resolving conflicts

We currently employ a simple conflict resolution policy which cancels the activation of a rule and provides the user with details about the conflict (line 11 in Fig. 3). We plan to extend this to support a richer set of conflict resolution policies, such as "keep existing", "replace with new", "remove both", "allow conflicts", as well as interaction with users.

V. Implementation Details

Making a request to activate a new rule triggers the process of verification which may detect and resolve conflicts or it may find no inconsistencies. Once checked, a new rule definition must be delivered to the appropriate enforcement point based on the contents of the subject, service and task definitions. After distribution, rules are translated into reconfiguration commands which are invoked on the system. Our prototype design is platform independent and can be easily extended to support a variety of platforms (e.g. CISCO IOS, DummyNet [8], NOX [9] [10] etc) rewriting only the interpreter module.

Currently, our prototype generates Linux "*tc*" [11] commands. The Linux traffic shaping subsystem, consists of three different elements: filters, classes and queuing disciplines [12]. Linux network administrators build their custom structures using these components aiming to achieve the desired shaping behaviours for various types of traffic. We employ the hierarchical token bucket (HTB) queuing discipline (*qdisc*) at the root of our outgoing interfaces for hosts. Filters redirect

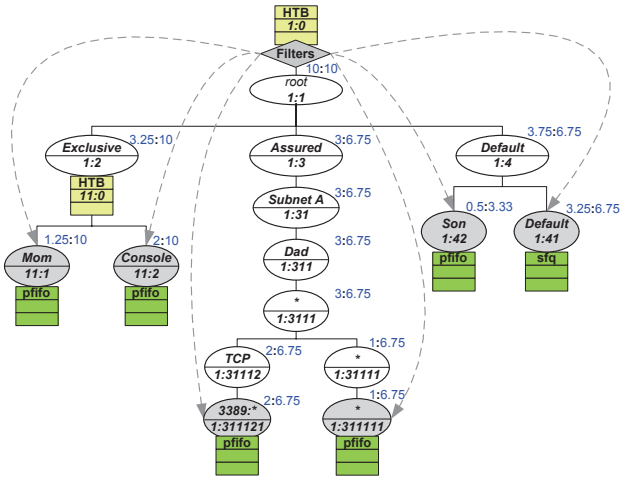


Fig. 7. The resulting Linux "tc" hierarchical queueing model

outgoing traffic to the appropriate queues, and HTB classes control in a hierarchical manner the usage of bandwidth resources from these queues. Unfortunately, the HTB has certain shortcomings that required us to create *qdisc* hierarchies which under certain circumstances could be slightly different from for the desired ones. For instance, in *tc* it is not possible to express only maximum cap rates but it is always required to specify an assured rate. Also, explicit class borrowing policies cannot be specified but instead, these policies are implied from the combination of hierarchy topology and rate/ceiling parameters.

In Fig. 7 we illustrate the resulting hierarchical *tc* queuing model given the tree structure in Fig. 6, for the home network case scenario we have used throughout the paper. In our example, we have also included an extra rule which assigns 33% of gateway's total downlink bandwidth (X) to all traffic towards $Dad's$ host. We have done so, in order to demonstrate the hierarchical sharing model in tc for assured rates. We assume that total bandwidth at gateway (X) is equal to 10 Mbps.

One of the key components in tc are filters because they redirect traffic to the appropriate classes. We filter traffic at the root of the tc model, based on the rule definitions contained in outer nodes $(R_1, R_2, R_3, R_4, R_5)$ of Fig. 6. Filters are applied once, and traffic can only be redirected to a specific class, which means that we need a mechanism to distinguish specific traffic from more generic. In order to achieve this we mark filters with priorities; the more generic the rules are, the lower the priority of their corresponding filter is. Using this prioritisation technique, we can classify traffic in a hierarchical manner.

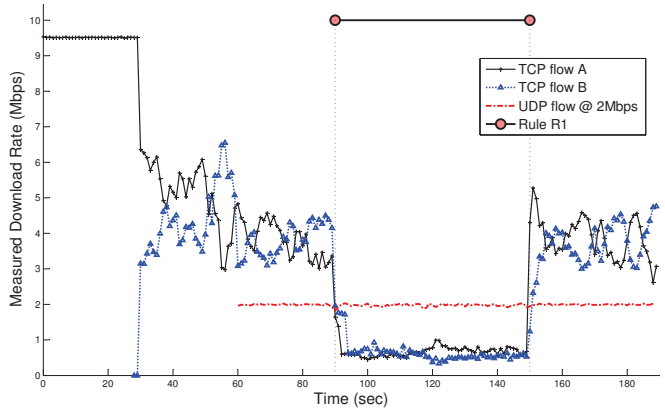
VI. Experimental evaluation

In this section we evaluate the accuracy and the consistency of our system. We aim to verify that the resulting behaviour of the network reflects the semantics of a sharing plan's contracts. We use "iperf" to generate artificial tcp and udp traffic and "tcptrace" to analyse the traffic at the flow level. To avoid the performance fluctuations induced by the wireless link, we rely solely on switched Ethernet. For the gateway we use an Asus WL-500gP flashed with OpenWrt (v10.03), and installed the modules for supporting hierarchical token bucket queue scheduling. The background traffic was directed to a separate client host in the local network.

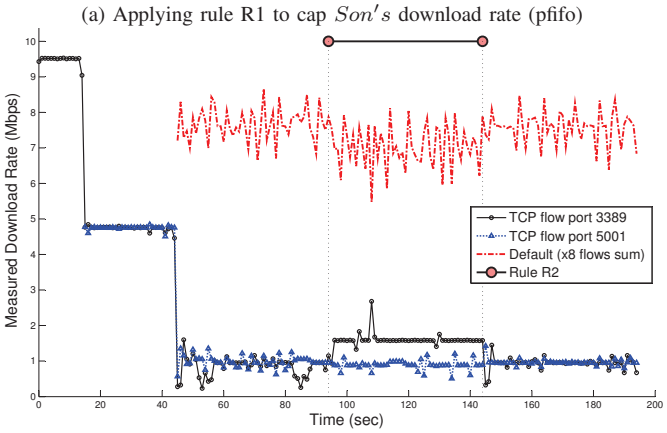
In Fig. 8a we show the result of enforcing rule R1, that limits the total download rate of $Son's$ host to $X/3$ ($\sim$ 3.5 Mpbs). All flows have the same destination (Son). Between seconds 90 and 150, where R1 is active, we observe a sharp drop in the rate of $Son's$ flows, which share the remaining resources left by the greedy *udp* flow. The fluctuations in the download rates, are mainly because we've used *pfifo* as our queuing mechanism, that serves traffic on a per-packet best effort basis. Over the long term, both flows download equal data volumes.

Our second experiment evaluates the correct enforcement of an assured rate contract. More specifically rule R2 (Fig. 8b), is active for 50 seconds (from 95 to 145) and assigns $Dad's$ host with a rate of $X/6$ ($\sim$ 1.6 Mbps). Note that the default (unmanaged) traffic consists of 8 *tcp* flows, which along with the two flows of $Dad's$ host share the available bandwidth on a fair basis (roughly 1 Mbps each). The extra 0.6 Mbps which gets the traffic to port 3389 during R2 is equally distributed among the rest 9 flows. This explains why the drop in the *tcp:5001* flow is not noticeable. Also, another interesting observation is that if stochastic fair queuing (*sfq*) is used, the bandwidth is equally assigned to each individual flow of the default traffic.

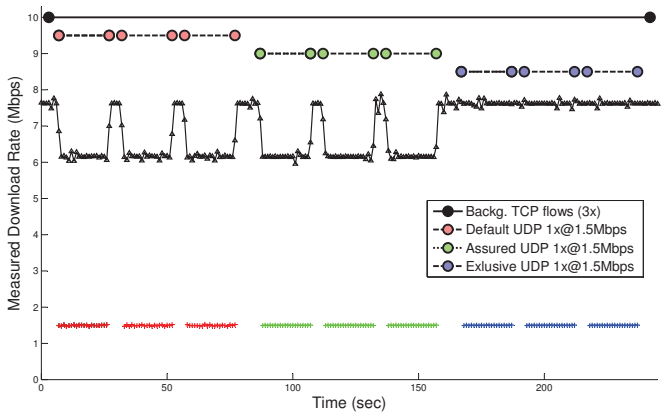
Bandwidth sharing contracts also have an impact on the mean queuing delays. This is because each class of traffic has its own queues, and thus, the packets belonging to an exclusive/assured class, tend to experience less delays in the presence of heavy cross traffic. Figure 8c describes the experimental scenario we've run for identifying the delay characteristics among the different classes of traffic. We have assigned "HostA" with an assured rate of 2 Mbps and an exclusive rate of 2 Mbps for traffic destined to ports 5001 and 5002 respectively. The background traffic to "HostB" is constant throughout the whole duration of the experiment (three *tcp* flows). For each class of traffic (Default[*udp:5000*], Assured[*udp:5001*], Exclusive[*udp:5002*]) towards "HostA" we run three rounds of *udp iperf* flows, lasting 20 seconds each. The results are shown Fig. 8d are averaged over the three rounds. The delay jitter is substantially lower for the assured and exclusive classes, but between them, the difference is nearly negligible. This is an artifact of Linux's *tc* design, because assured rates have only a small initial delay when the HTB class has lent transmission tokens to another class (probably the default one). After that initial delay, the jitter characteristics are similar to the that of the exclusive class. Finally, at second 160, the default traffic's download rate seems not to be affected by the *udp* flows. This happens because default and exclusive classes do not share common resources: the exclusively assigned 2 Mbps stays unused until second 160.



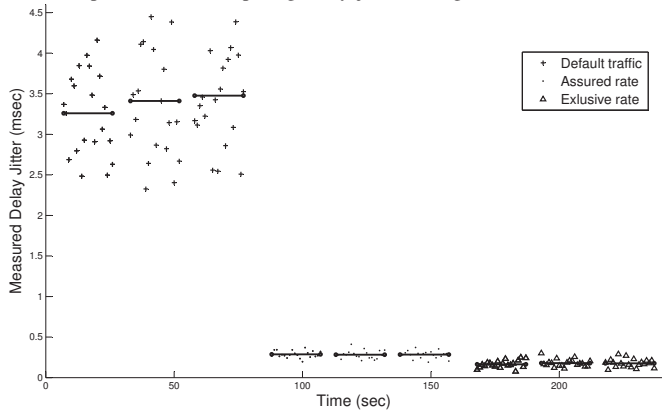
(a) Applying rule R1 to cap $Son's$ download rate (pfifo)



(b) Applying rule R2 to assign $Dad's$ host with assured rate



(c) Experiment for comparing delay jitter among different contracts



(d) Measured delay jitter for traffic under different contracts

Fig. 8. The effect of sharing plan rules on bandwidth resources

VII. Related Work

Home network management is starting to draw the attention of the research community, both for infrastructure support and HCI [13]. [14] and [15] aim to facilitate management tasks through smart service discovery. They adopt OSGi [16] but don't support the dynamic aspects of home networking like bandwidth sharing. In [17], [18] a policy based architecture is proposed to facilitate quality of service extensions and security management, but lacks an implementation. An elegant information plane architecture is used in [19] to support management and novel new user interfaces. The architecture uses a high-performance stream database, and could readily be used to trigger our bandwidth sharing plans. A very different approach is proposed in [20] where management tasks are outsourced to cloud services. This has potential, but we think that local and ISP-level management will be more effective in the short-term.

In the context of bandwidth management, HomeMaestro [21] advocates the notion of "application fairness" based on a user feedback and application-specific weights. In [22] a dynamic resource allocation scheme is described that tries to optimize allocations across wireless and wired devices in a weighted fair manner. As in HomeMaestro, applications may have different weights, and based on this prioritisation scheme, the system performs host coordinated rate control to optimize the network's overall performance. These approaches complement our goals by introducing the ability to prioritize traffic, although their models do not reflect network's overall hierarchy and management actions are only applied on a per-connection (and per-application) basis. Finally, finite state machines are used in [23] to enforce low-level adaptive network behaviors such as wireless channel selection to mitigate interference and increase capacity.

VIII. Summary

There is a need in many homes to be able to define a bandwidth sharing plan for the users, services and applications of the home network. This is a difficult task requiring abstractions and user interfaces that are easy to understand by home users. A key aspect for any home management system supporting bandwidth sharing is to be able detect conflicts and errors in plans. Such verification can be used to inform the user and/or trigger a resolution strategy. In this paper, we have described our approach for modelling and for detecting conflicts and errors in bandwidth sharing plans consisting of a number of bandwidth brokering rules. If no errors are detected, our bandwidth brokering rules are translated to system specific traffic shaping commands, and enforced on appropriate devices in the network. For our future work we are looking at integrating with a novel new user interface from the University of Nottingham based Comic Strips and deploying the resulting system in a number of real homes.

ACKNOWLEDGMENT

This research was supported by UK EPSRC research grant EP/F06446/1 (Homework).

REFERENCES

[1] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "The impact and implications of the growth in residential user-to-user traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 207–218, August 2006. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159938

[2] "Uk broadband speeds: The performance of fixed-line broadband delivered to uk residential consumers," Ofcom, Tech. Rep., 2010.

[3] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 43–56. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298313

[4] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 90–102. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644904

[5] K. Papagiannaki, M. Yarvis, and W. S. Conner, "Experimental characterization of home wireless networks and design implications," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–13.

[6] X. Xing and S. Mishra, "Where is the tight link in a home wireless broadband environment?" in *Proc. IEEE Int. Symp. Modeling, Analysis & Simulation of Computer and Telecommunication Systems MASCOTS '09*, 2009, pp. 1–10.

[7] N. Hu, L. Li, Z. Mao, P. Steenkiste, and J. Wang, "A measurement study of internet bottlenecks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, pp. 1689 – 1700 vol. 3, march 2005.

[8] M. Carbone and L. Rizzo, "Dummynet revisited," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 12–20, April 2010. [Online]. Available: http://doi.acm.org/10.1145/1764873.1764876

[9] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 105–110, July 2008. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[11] W. Almesberger, E. Ica, J. H. Salim, A. Kuznetsov, and I. Moscow, "Differentiated services on linux," in *in GLOBECOM: General Conference*, 1999, pp. 831–836.

[12] B. Hubert, T. Graf, G. Maxwell, M. Oosterhout, P. Schroeder, J. Spaans, and P. Larroy, "Linux advanced routing and traffic control howto," Lartc, Tech. Rep., 2005.

[13] J. Yang, W. K. Edwards, and D. Haslem, "Eden: supporting home network management through interactive visual tools," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ser. UIST '10. New York, NY, USA: ACM, 2010, pp. 109–118. [Online]. Available: http://doi.acm.org/10.1145/1866029.1866049

[14] S. Zeadally and P. Kubher, "Internet access to heterogeneous home area network devices with an osgi-based residential gateway," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 3, pp. 48–56, December 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1356244.1356249

[15] P. Bull and M. Harrison, "Managing broadband home networks," *BT Technology Journal*, vol. 24, pp. 79–85, 2006, 10.1007/s10550-006-0023-z. [Online]. Available: http://dx.doi.org/10.1007/s10550-006-0023-z

[16] OSGi Alliance. (2007) Osgi service platform release 4. [Online]. Available: http://www.osgi.org/Main/HomePage. [Accessed: Jun. 17, 2009].

[17] A. Rana and M. Foghlu, "New role of policy-based management in home area networks - concepts, constraints and challenges," in *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, 2009, pp. 1 –6.

[18] A. I. Rana and M. O. Foghlú, "Policy-based network management in home area networks: interim test results," in *Proceedings of the 3rd international conference on New technologies, mobility and security*, ser. NTMS'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 334–336. [Online]. Available: http://portal.acm.org/citation.cfm?id=1790343.1790407

[19] J. Sventek, A. Koliousis, N. Dulay, D. Pediaditakis, T. Rodden, T. Lodge, O. Sharma, M. Sloman, B. Bedwell, K. Glover, and M. Richard, "An information plane architecture supporting home network management," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management*, 2011.

[20] C. Gkantsidis and H. Ballani, "Network management as a service," Microsoft Research, Technical Report MSR-TR-2010-83, 2010.

[21] T. Karagiannis, E. Athanasopoulos, G. Christos, and P. Key, "Homemaestro: Order from chaos in home networks," Microsoft Research, Technical Report MSR-TR-2008-84, 2008.

[22] C. Gkantsidis, T. Karagiannis, P. Key, B. Radunovic, E. Raftopoulos, and D. Manjunath, "make a wireless networks," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 265–276. [Online]. Available: http://doi.acm.org/10.1145/1658939.1658970

[23] D. Pediaditakis, L. Mostarda, C. Dong, and N. Dulay, "Policies for self tuning home networks," in *Proceedings of the 10th IEEE international conference on Policies for distributed systems and networks*, ser. POLICY'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 29–32. [Online]. Available: http://portal.acm.org/citation.cfm?id=1812664.1812671