Segmented Adaptation of Traffic Aggregates

Hermann de Meer and Piers O'Hanlon

Dept. of Computer Science, University College London
{H.DeMeer,P.OHanlon}@cs.ucl.ac.uk

Abstract. Congestion control in heterogeneous Quality-of-Service (QoS) architectures remains a major challenge. The solution proposed in this paper entails three constituents. Taking the current trend towards Differentiated Services (DiffServ) as a likely candidate for future Internet QoS-architectures, the three constituent parts of this approach are based on aggregated congestion control, domain-based congestion control, and class-of-service based congestion control. Generally, the overall framework for congestion control, as suggested here, reflects essential properties of underlying QoS-architectures and their instantiations in real implementations. As such an approach calls for highly flexible architectures, we suggest the use of Active Networking, and in particular Application Level Active Networking, as an enabling technology for a seamless and rapid integration of the proposed scheme into current architectures.

1 Introduction

Congestion control in IP networks has traditionally relied on mechanisms of the transport protocol TCP, and thus, on the cooperation of applications in the face of network congestion. With the advance of real-time multimedia communication applications on the Internet, on the other hand, a non-congestion sensitive usage of UDP has become increasingly popular for several reasons. The prospect of a massive usage of non-responsive transport protocols such as raw UDP, however, has created some concern about the viability of a "laissez-faire" approach that has traditionally been adhered to in the Internet [3].

In the same vein, issues in sharing bottlenecks and techniques of how to enforce fairness among competing elastic and non-elastic applications has triggered a hefty discussion within the Internet research community recently. While some form of fairness among competing flows appears appealing to one camp of researchers others are worried about the implied overhead of policing and enforcing control policies on a flow basis. To yet another camp, the idea of enforcing fairness in sharing bottlenecks does not appear as compelling at all. Disregarding implementation and run-time overhead, a fair sharing of bandwidth may not easily translate to a concept of fairness on the application level as QoS requirements and pricing conditions may vary vastly. While the concern about congestion control is being widely shared the jury is still out on measures of how to approach the problem [3].

QoS architectures within the confines of the DiffServ model are currently being introduced into the Internet to provide preferred handling of privileged traffic load classes [8]. Subscribers to such a privileged class expect to receive a higher level of service at a premium price. Within the confines of a DiffServ architecture, and in particular within Assured Forwarding (AF) subclasses, congestion may still arise and QoS violations may occur at times, despite traffic conditioning and resource provisioning. QoS architectures rely on class-dedicated resource provisioning of some sort and thus try to reduce the *probability of congestion occurrence* on the one hand and the *impact of congestion events* on privileged traffic on the other hand. Although the latter aspect may seem less obvious, we believe that it does reveal a most important aspect of the semantics of QoS architectures. Our approach is essentially based upon that observation.

Congestion can be seen as a possible cause (error) or "fault" in delivering a networking service with a required QoS. Since the possibility of such a QoS fault is likely to be an occasional matter-of-fact characteristic of any DiffServ architecture, we suggest to systematically take that into account as part of an overall model of QoS semantics of a given architecture. Such an approach is common practice in designing fault-tolerant distributed systems [2]. We therefore advocate to characterize QoS architectures not only in terms of assured transport privileges, but also in terms of adequate recovery measures in the presence of QoS faults as a result of congestion.

Assessing current trends in the design of QoS DiffServ architectures reveals inherent possibilities of QoS faults. Taking the derived fault models into account, we conclude the measures as presented in this paper. The suggested solution is based on the three elements of *aggregated congestion control*, *domain-based congestion control*, and *class-of-service based congestion control*. The presented approach is intended to contribute to the solution of some of the pending congestion control problems within the confines of QoS architectures as stated above.

Our approach is based on the introduction of new control and management functions into existing architectures. Introducing new services into an existing networking environment, however, is a challenge of its own, as re-programming of routers has traditionally been solely controlled by hardware vendors. While Active and Programmable Networking technologies have been pursued as a possible once-and-for-all solution to the often lamented unflexibility of networking infrastructure by the research community, there is still a long way to go for Active Networking infrastructure to be widely available [1]. Our pragmatic solution, therefore, relies on our Application Level Active Networking (ALAN) infrastructure, called FunnelWeb, that requires only a minimum support from the networking layer [4]. The ALAN approach avoids many performance and security risks that have often been considered as a barrier to more general acceptance of Active Networks.

In Sect. 2, we first introduce the notion of QoS fault-tolerance measures in DiffServ architectures and then look at segmented adaptation as a special case of it. An illustrating test scenario completes that section. FunnelWeb, our Application Level Active Networking infrastructure, is introduced in Sect. 3, followed by presentation of an implementation of the segmented adaptation scheme based on that infrastructure. Section 4, which shows some of our simulation studies and first numerical results, is followed by the conclusion in Sect. 5.

2 Decomposition of Elasticity and Feedback

2.1 QoS Failure Semantics in Networks and Segments

Current discussions on DiffServ QoS architecting largely fall into two categories. One focus is on traffic conditioning and admission control at network edges and the other is on resource provisioning at the network nodes by defining so-called per hop behaviors (PHB). Combining proper traffic conditioning at the edges with nodes configured according to PHB specifications is designed to result in well-defined *edge-to-edge behaviors* for an *aggregate* of traffic load. Such a behavior aggregate, whose scope is limited to an administrative domain or an autonomous system (AS) as a *segment* of the network, is referred to as a per domain behavior (PDB) [6].

The ultimate goal in the current DiffServ bottom-up approach is to compose end-to-end QoS services from PDBs. Currently several variants of PDB definitions are underway within the IETF. Conditioning and provisioning are performed based on service level specifications (SLS) as part of more general service level agreements (SLA). In addition to technical parameters and values of an SLS, characterizing the service as delivered by a network segment, an SLA may contain pricing and other contractual features.

A wide spread belief is that adequately provisioned resources at the network nodes and carefully set conditioning elements at the edges can assure, or even guarantee, a targeted level of QoS. While that may be true for many conditions, there have not yet been any proofs of such an assignment procedure to reliably exist under fairly general conditions. Provisioning and conditioning are rather static and slow operations that may need to anticipate traffic matrices in a reasonably general form, need to account for repeated occurrences of aggregation/disaggregations at nodes of the particular network segment, or to possibly deal with other hard to control stochastic events that may have an adverse impact on the delivered QoS of the network segment. While the likelihood of occurrence of congestion inducing events that may trigger a segmented QoS fault may be low, we nevertheless argue that precautions should be taken against them. This is particularly the case given the hierarchical structure of the DiffServ architecture in which end-to-end services may well be composed of a chain or mesh of PDBs. While a segmented QoS fault may be rare under ideal conditions, the whole chain could break if any segment exhibits a QoS failure, which is likely to occur an order of magnitude more often than a QoS failure of a single segment.

As a result of the aforementioned analysis, we argue for the explicit extension of the QoS DiffServ model to incorporate (segmented) QoS fault tolerance. Our view is that this is in compliance with Geoff Huston's recent proposal for the next generation of DiffServ architectures [9]. Only we are suggesting a more systematic approach based on the well-understood technology of distributed fault-tolerant systems [2] and its application to the concept of (segmented) QoS faults. Expressing Geoff Huston's proposal in terms of these concepts and terminologies it could be rephrased as augmenting PDBs with QoS error state detection mechanisms in order to signal implied QoS fault events to the edge routers. The edge routers in turn may choose to trigger error recovery techniques in order to re-establish a well-defined QoS level according to a given SLA. The recovery action to perform would clearly depend on the QoS class itself as well as on other factors such as pricing. For graceful adaptation we suggest the introduction of *service curves* for definition of QoS levels within the SLA. The benefit of such an approach is that QoS guarantees can be expressed as simple functions of the service curves, as opposed to rigid definitions in traditional SLAs. This technique would allow for smooth movement between different service levels dependent upon network state and stipulated class of service through the selection of appropriate service curves. A simple example would be to dynamically reduce the throughput of a targeted QoS class as opposed to retaining throughput but compromising QoS measures such as delay or jitter for the whole class.

In general, any disturbances could lead to an error such as congestion. Congestion, in turn, may lead to a violation of the targeted QoS, which is referred to as a QoS fault in the particular segment. Local error recovery methods such as adaptive QoS routing could mask the fault so that no QoS failure of the PDB becomes externally visible. But error recovery could also, and mostly will, rely on external cooperation as provided by contracted elasticity of up-stream domains to *adapt* the service-qualified load sensibly. Formally, the purpose of introducing QoS fault tolerance is to reduce to likelihood of "externally" visible end-to-end QoS failures or to classify failures if fault cannot be tolerated. For more information on the usage of the terms "error", "fault" and "failure" within the confines of distributed fault-tolerant systems refer, for example, to [2].

QoS error recovery actions would be performed on whole traffic *aggregates* in accordance with the DiffServ model and are suggested to be specified as part of (bilateral) SLAs among providers. No involvement of any flow-specific knowledge or measure is needed or wanted for the method to be effective. We see this as an argument in favor of scalability and flexibility. In analogy to TCP-based congestion control, one could refer to cooperating service providers as being *elastic.* We believe elasticity and cooperation among service providers in terms of realizing QoS fault-tolerance as vital for the success of the DiffServ model, in analogy to the viability of the best-effort model based on elasticity or TCP-friendliness of cooperating applications.

Once the DiffServ model has been adopted, we believe that the approach of segmented adaptation and other domain based error recovery methods as suggested here do not imply any further violations of the end-to-end principles [7]. On the contrary, incorporating QoS fault tolerance into the DiffServ model increases flexibility and mutual independence between applications and the network transport mechanisms as it re-establishes backward compatibility with TCP-like end-to-end congestion control [3]. It provides the means for self-protection of network segments and may even allow the introduction of *incentives* to applications for backing off, perhaps based on local access policies, by a customized reduction in advertised bandwidth. We see segmented QoS error recovery as an essential means to enable the network to control its own resource utilization within the DiffServ framework. Although elasticity of applications does help significantly, the network should not *rely* on applications to cooperate in order to deliver its advertised services.

Our approach introduces a dynamic factor into SLAs and the execution of management tasks and traffic conditioning operations. Currently we limit ourselves to implementing simple QoS fault indication mechanisms and restrict the experimental setting to dynamically reconfiguring traffic conditioning operations to yield an adaptation effect of traffic aggregates. The concept, however, is by no means limited to these specific measures and is potentially open to any error state (or congestion) discovery, QoS fault notification, and QoS error recovery procedure. The approach is also backward compatible to application level congestion control mechanisms that can (and should) be run concurrently. In contrast, network segments can protect themselves from non-cooperative applications by enforcing a network segment based dynamic adaptation of traffic aggregates.

Depending on local policies and effectiveness of QoS error recovery, congestion signals may, however, propagate all the way back to access domains. Local access policies and strategies may become effective and non-elastic applications can, for example, be penalized. But these decisions would be made locally by the access bandwidth brokers and would not impose any burden on the core or edge routers further down-stream.

The way adaptation of traffic aggregates is performed, as well as other mechanisms forming part of a QoS fault tolerance framework, also depends on the service class that is supported by a PDB in a given segment. This entails error detection schemes such as random early detection (RED) [3], QoS fault classification mechanisms such as identifying affected and responsible traffic aggregates, and also the adaptation strategy itself. A best-effort service, for instance, offers little clue about the nature, scope and duration of a congestion state. TCP is a viable control mechanism for elastic applications, which oscillates to sense bottlenecks in short time scales. Entropy is reduced, however, in the case of DiffServ service classes. Here the occurrence of congestion may indicate longer term provisioning problems rather than coincidental short bursts that have an adverse effect. Adaptation may occur rarely but load reduction may have to last longer under these circumstances. We therefore anticipate a time scale for these slower adaptation steps to be in the order of session durations for Assured Service traffic aggregates as session having an impact on the traffic matrix. Expedited Service classes my require even longer down scaling of the traffic aggregate, perhaps permanently so, as a congestion state could be seen as a severe error that indicates a prohibitively poor provisioning strategy. While it seems to be obvious that service-class types have a significant impact on the best adaptation strategy for traffic aggregates, we are far from having final answers to these questions at present time. We consider this for further research but regard a *differenti-*

ated adaptation behavior that reflects service class differentiation as an essential constituent for segmented adaptation of traffic aggregates in the DiffServ framework.

2.2 An Economical Motivation

A definition of suitable SLAs including QoS fault tolerance mechanisms would be subject to *economically motivated* bilateral agreements and *technologically enabled* solutions. For example, imagine an up-stream provider having a contract with a down-stream provider to forward 5Mbps of a certain Assured Forwarding (AF) QoS-type of traffic aggregate through its domain. Minimizing the percentile of QoS failures by overprovisioning the down-stream networking segment may get excessively costly beyond a certain percentile threshold. In contrast, it might be much less expensive, and thus to the mutual benefit, to agree on clearly stated QoS fault models, fault notifications and recovery procedures. The resulting QoS fault tolerance may involve cooperation between neighbouring providers such as reducing the aggregated rate to 2Mpbs, say, that is sent down-stream upon receiving a congestion signal. Nothing is said of how the up-stream provider does achieve the requested QoS recovery measure, be it by re-setting traffic conditioning parameters at its corresponding egress link or by applying QoS routing and traffic management techniques, or any combination thereof and further measures. The down-stream provider would rely on the potential - and rare - case of cooperative elasticity in the way it manages and runs its own network segment. Of course, the necessary QoS fault-tolerance mechanisms have to be in place as well, which is a cost factor too, so a trade-off analysis may reveal the optimum point of operation based on all given local conditions and QoS requirements.

Specifying QoS failure semantics and provisioning of QoS fault tolerance based on bilateral agreements may not only enhance the end-to-end QoS as to be experienced by applications but is also envisioned to lead to a mode of operation that allows minimization of constraining cost functions.

2.3 A Test Scenario

A test scenario is given in Fig. 1. Four autonomous systems (AS) are interconnected by properly defined DiffServ links with the capacity as indicated in the legend of Fig. 1. All indicated intra-AS links are also considered to be DiffServ capable. Edge routers are labeled by names with initial letter E, followed by a number indicating the particular AS the router belongs to and a symbol indicating the particular router in that domain. Core routers are indicated by an initial letter C. Hosts are simply designated by the source running on it.

Assured Forwarding (AF) links have been set up between the nodes with the edge routers configured for time sliding window with three color marker (TSW3CM) policier. The initial state is that four hosts are transmitting traffic. One FTP source is running over a TCP connection from end host *src1* in AS1 to its sink in AS3. One 4Mbps UDP constant bit rate source (with randomized departure times to avoid phase effects) is running from *src2* in AS4 to its sink in AS3. Similarly, *src3* in AS4 is transmitting a 2Mbps CBR stream to its sink in AS3. And, finally, a third CBR UDP source *src4* is transmitting 4Mbps UDP traffic to its sink in AS3.

We have artificially created the congested AF link c3r1–c3r2 in AS3 by defining the traffic matrix and choosing the link capacities accordingly. For the sake of demonstration we have naturally exaggerated the congestion event as we are not particularly interested in an exact modeling of congestion occurrence at this point but in the effectiveness of our congestion recovery methods in relation to a given AS.

Congestion Notification (CN) signals originating from link c3r1–c3r2 in AS3 are propagated back to the edge routers and adaptation of traffic aggregates is performed to re-condition the traffic load that enters AS3 at the particular ingress link. Two solutions are possible in this case, either the advertised capacity of egress link e2r3–e3r1 or the advertised capacity of ingress link e3r1–c31 can be reduced accordingly. As results presented later suggest, the latter action is considered as less desirable and should only be used as a intermediate step for self-protection of AS3. The goal, however, is to initiate a reduced traffic flow from AS2 to AS3 with respect to a congested traffic class. This is considered a method to recover from a segment-AS3 QoS-fault and is achieved by providing elasticity of AS2.

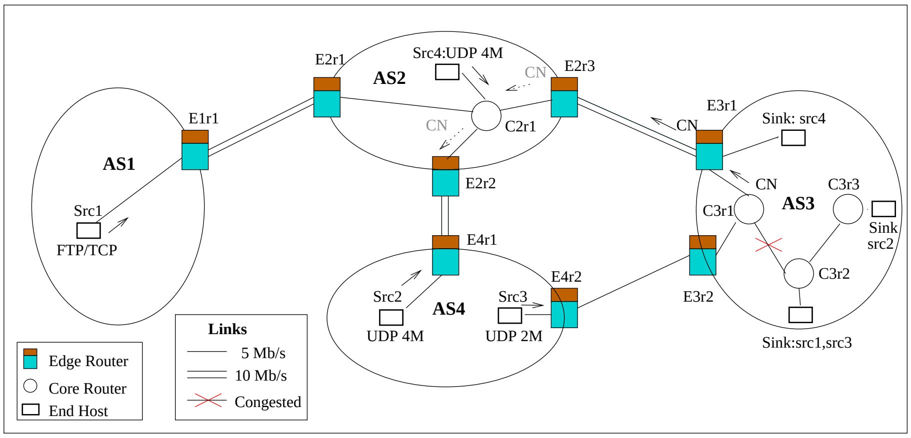

Fig. 1. Test Scenario

Once adaptation of the traffic aggregate on the identified ingress link of AS3 has been performed properly, AS3 should be well protected and be back normal operation of Assured Forwarding type of service without losses at core nodes. The edge router of AS2 may deside to forward the CN signal to AS4 in order to trigger a reduction of advertised AF capacity of link e4r1–e2r2. If that step is performed properly, both AS2 and AS3 should have recovered from a possible segmented QoS failure and be back to normal AF forwarding mode, albeit at a potentially reduced rate. It is up to AS4 to decide on quenching source src2 or on taking other recovery actions internally. Of course, the scenario indicated in Fig. 1 is likely to be a partial representation only with many more links and nodes available in reality, providing more options for recovery.

Given that aggregated links may be shared by many traffic types in DiffServ we would like to mention the fact that the presented scenario implicitly entails QoS routing as flows *src2* and *src3* follow different paths, although originating and terminating in the same ASs. While not particularly significant in the results presented in this paper, our overall approach does take QoS routing into account. We will not further elaborate on QoS routing in this paper, however.

3 An Implementation Based on ALAN

3.1 The Concept of ALAN

The concept of Application Level Active Networking (ALAN) has been developed to introduce flexibility and openness into networking architectures, at the application level, without compromising performance and security of core networking functions [4].

ALAN differs from network layer Active Networking (AN), where packets carry active code which may be executed at the network level in devices such as routers. The AN approach attempts to solve the flexibility problem while creating a number of new challenges, in particular, with respect to security, network stability and performance. In contrast, the ALAN approach leaves the basic network infrastructure unchanged and provides a framework for deployment of application level active services. General purpose computational nodes are placed inside the network at strategic locations to host execution environments that can be dynamically installed to perform application level functions that, however, interfere in a very limited way with network transport mechanisms. This concept is similar to current methods in optimization of content delivery in the Internet that have been widely adopted by the industry. The main difference is that we are deploying general purpose "boxes" that are designed and intended to be easily re-programmable. For the time being, we believe it much more likely that ISPs will tolerate, or even support, placement of these "boxes" in their domain rather than supporting "real" Active Networking.

3.2 FunnelWeb

We are using the *FunnelWeb* [4] implementation for ALAN - see Fig. 2. This system consists of deploying active elements in the network which provide an application level execution environment. In FunnelWeb the active applications, or active services, take the form of *proxylets*. Proxylets are written in Java and are loaded by reference. Proxylets may be loaded and executed on active node machines running an *Execution Environment for Proxylets* (EEP). There are interfaces for monitoring and control of the proxylet on the EEP.

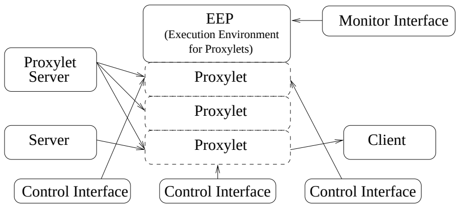

Fig. 2. FunnelWeb Architecture

A mesh of EEPs is initially bootstrapped and maintained on the principle of self-organization. As a result, EEPs can be dynamically added to strategic locations inside the network or removed if required. Self-organizing application layer routing (SOAR) is being used to form a hierarchical topology. Related proxylets running on different EEPs form a self-organizing distributed service themselves by exploiting SOAR in turn for establishing and maintaining connectivity. More details on the infrastructure, however, are beyond the scope of this paper and can be found in [4].

3.3 Active Segmented Adaptation

In our approach, adaptation of inter domain traffic is to be performed at the edge routers of autonomous domains according to pre-negotiated, bi-lateral, service curve oriented SLA. Normally, traffic crossing network boundaries is handled within negotiated standard SLS operation. An SLS may specify what to do in the case of out-of-profile traffic [10], resulting in some level of traffic conditioning at the edges such as dropping or re-classification. In contrast, dynamic adaptation of traffic aggregates may well be performed on *in-profile* traffic in the *exceptional case* of congestion events. Adaptation may be implemented in any form that reduces the relevant traffic load class, be it by dropping, re-classification or re-shaping. The resulting action is dependent upon the local and class-based QoS failure semantics, the correspondingly suggested recovery methods, and further conditions such as pricing that may be included in a bi-lateral SLA.

Active services in the form of proxylets are dynamically deployed to augment and customize traffic conditioning at the edges. We refer to this approach as an active bandwidth broker (ABB). An ABB also forwards and filters congestion signals to neighbouring domains according to actively extended SLAs. A dynamic change of traffic conditioning can be accomplished easily as the required traffic conditioning modules are already in place due to the underlying Diff-Serv architecture. Only an interface is needed for a dynamic parameterization of the conditioning modules. The change of the parameter settings is triggered by dynamically attached proxylets. Local policies can easily be incorporated and changes to policies can be realized *on demand* due to the inherent flexibilty of FunnelWeb.

Segmented adaptation is designed to utilize explicit CN's from the core of an AS to its edges. These congestion signals may be generated when certain resource utilization levels at some router exceed given thresholds. Such notifications contain information from packet headers regarding the sources of congestion. At the current point we plan to use multicast notifications from the congested router to the proxylets attached to the edge routers. It may also be possible to unicast such notifications by usage of an active intermediary that would be able to specifically route the CN signal.

Once the proxylets at the edge router, as a constituent of an ABB, receives a CN they may decide to comply with it or to discard it, depending on the valid policy. Type of action taken may include a variety of strategies to control aggregates' behavior dependent upon the nature of the congestion and the service class affected. In addition, the ABB may choose to forward the CN to peering ABBs and to trigger a corresponding adaptation of the traffic aggregate at corresponding actively extended edge routers (potentially more responsible for the congesting traffic) dependent upon the contents of the CN and installed policy. Action taken would have a lifetime dependent upon the continuing congestion state of the core network as reported by subsequent refreshing CNs. If neither CN signals are received from the defective down-stream domain nor any arriving traffic exceeds the lowered egress rate after a "reasonable" period, recovery from an AS QoS fault is assumed to be completed and the link is opened again to full capacity. Referring to Fig. 1, when recovery has been completed for AS3 and AS2 only link e4r1–e2r2 remains scaled down as long as recovery has not been completed within AS4, all other AF links are back to normal. Note that this is the current procedure of AF adaptation based on intuition, but more systematic, preferably formal modeling, studies are needed.

We are implementing the system using two proxylets as illustrated in Fig. 3.
- The *SLA proxylet* provides facilities for implementation of SLS service curve behaviour which modifies the parameters in the conditioning elements. The SLA proxylet also specifies the conditions in which peering ABBs should get involved and the congestion notification signal should propagate further up-stream. The *ASA proxylet*, where ASA refers to active segmented adaptation, controls operation of QoS error recovery in terms of aggregated adaptation as discussed earlier.

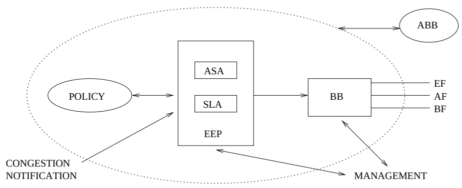

Fig. 3. Active Bandwidth Broker

Using FunnelWeb provides high flexibility to dynamically change policies or to update adaptation and other recovery procedures upon demand. This could be useful in practice if SLAs are required to be changed more frequently among providers. FunnelWeb could be considered as a general purpose enabling technology for introducing flexibility into QoS DiffServ architectures. In our case, FunnelWeb allows us to rapidly and seamlessly integrate SLA extensions into existing networking infrastructures to create our customized testbed and to experiment with it. Given sufficient motivation the proxylets could be replaced by fixed hardware and software components that perform the same functions more efficiently. In that respect, FunnelWeb may be considered as an enabling technology for rapid prototyping.

4 Some Simulation Results

Referring to the test scenario in Fig. 1, we performed some early simulations to study the effectiveness of adaptation of traffic aggregates. The simulations have been carried out using The Network Simulator [5]. While our focus is on inter-domain adaptation of traffic aggregates, we also present additional results on intra-domain aggregated adaptations on links local to AS3. This provides more systematic insights into the impact of adaptation steps. Indeed, the concept would also allow adaptations on local links as an immediate recovery method. Figure ?? depicts the flow specific effective throughput in the initial state of the network with the indicated congested link in AS3 of Fig. 1. Src1, src2, and src3 share the congested link. As a result, the TCP flow is almost completely starved while src2 and src3 share the remaining bandwidth but both suffer from considerable losses. Upon reception of CN signals, the total bandwidth on ingress link e3r1–c3r1 is immediately reduced to 2Mbps, shifting the congestion towards edge router e3r3. Such a measure could be useful as an immediate measure to shift a bottleneck away from a hot spot to increase the overall healthiness of a domain. The resulting situation can be seen in the left part of Fig. 4. The TCP session is still completely starved but at least the QoS of src3 is back to normal and the throughput is back to the targeted rate, link c3r1–c3r2 has recovered from congestion state and AF QoS is guaranteed again for that part of segment AS3. In a practical setting, this step could be seen as a realization of self-protection of the congestion affected AS.

Next, the CN signal is propagated further up-stream to edge router e2r3 in domain AS2. At that point the advertised bandwidth of link e2r3–e3r1 is reduced until no more congestion signals arrive. Domain AS3 now has completely recovered and is again fully complying with a PDB specification of the AF QoS class. Note, the approach does not boil down to a hop-by-hop control mechanisms, as may be wrongly concluded from the simple scenario. Only edge routers are involved in the adaptation process.

Finally, link capacity e4r1–e2r2 is reduced to 2Mbps and all other link restrictions are released shortly afterwards, as there is neither congestion observed on any link of AS2 or AS3, which have both completely recovered. The resulting effective throughput is given in the right part of Fig. 4. AS2 and AS3 are not only loss free and fully back to AF forwarding mode, but TCP src1 has gained it's share of link c3r1–3r2.

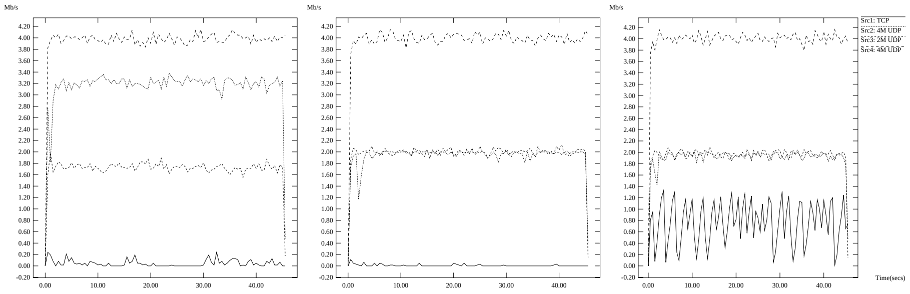

Fig. 4. Initial situation e3r1–c3r1:down2 e4r1–e2r2:down2

From the previous simulation studies it is rather obvious how difficult it appears to specify adaptation strategies for traffic aggregates reasonably well. So far we have only relied on our intuition, but more systematic investigations are obviously needed here. We like to remark, however, that the problem of defining strategies in back-propagation of CN signals and performing up- or downscaling of QoS DiffServ virtual links appears similar in nature to matching of traffic conditioning and resource provisioning in general. Hence, this problem must be solved for the DiffServ model to become viable in all possible means [9]. Once there are reasonable solutions for the provisioning/conditioning problem, these results should also be applicable to the related adaptation problem of traffic aggregates. But as we doubt there will ever be a perfect solution to the provisioning/conditioning problem, segmented QoS fault tolerance and adaptation of traffic aggregates may be as essential to the the viability of the DiffServ model, as the solution of the provisioning/conditioning problem itself.

5 Conclusions

In this paper we have introduced the concept of segmented adaptation of traffic aggregates. The concept reflects essential characteristics of evolving DiffServ QoS architectures such as those being built on the principle of composing end-to-end QoS services from autonomous segments that are characterized by their per domain behaviors (PDB). The entities of interest are traffic aggregates of given quality and quantity, which are negotiated between neighbouring providers who autonomously operate and control their domains. An adaptation of traffic aggregates is motivated by the observation that PDBs are likely to be defective with respect to the delivered service quality at times. Segmented QoS fault tolerance mechanisms, of which adaptation of traffic aggregates is designed to be a particular case, are to be provided to enhance end-to-end QoS and to confine or to reduce the probability of occurrences of end-to-end QoS failures. We envision segmented QoS fault tolerance as being the necessary "glue" to form end-to-end QoS services from well-defined PDBs as building blocks.

Our approach has been built on an extended concept of distributed bandwidth brokers that control the adaptation of traffic aggregates according to policies provided by SLAs. An actively extended SLA specifies how service-class dependent parameters and values of an SLS are altered upon reception of a congestion signal. As a result, the advertised bandwidth at an egress link of an up-stream edge router may be lowered. It is up to the up-stream bandwidth broker to push the congestion control signal further up-stream or to mask the QoS failure of the down-stream segment by local management operations such as by using QoS routing mechanisms. Ultimately, a congestion control signal *may* reach one or more access domains so that some applications could face pressure to adapt accordingly.

Building adaptation on edge-to-edge segmentation and aggregation of traffic is believed to result in a highly scalable approach for congestion control, or, more generally, QoS fault tolerance. Decoupling applications and network transport mechanisms further within the DiffServ model is likely to increase the overall flexibility in terms of the end-to-end argument. The introduction of network self-protection as an immediate consequence of the approach eliminates the reliance on the cooperation of applications in the face of congestion, whilst being fully backward compatible with TCP-friendly schemes for congestion control.

As a proof of concept, we are working on an implementation of a testbed, based on our ALAN platform that allows the introduction of new services into an existing networking infrastructure. While early simulation results on the effectiveness of segmented adaptation of traffic aggregates have been discussed in this paper, there is large scope for exploration of the area. Open problems can be identified in the segmented QoS fault tolerance problem space. Some examples of open issues are finding reasonably good procedures and parameter settings for error (congestion) detection, identification and classification of segmented QoS faults of traffic aggregates or defining service-class specific adaptation procedures properly. Other pending problems are investigating effectiveness of combining adaptation of traffic aggregates with other segmented QoS fault masking techniques such as QoS routing, or defining strategies for propagating congestion signals further up-stream. Each mentioned aspect opens up a new potential for research within the framework of segmented QoS fault tolerance, in general, and segmented adaptation of traffic aggregates, in particular.

References

1. Campbell, A.T., H. De Meer, M.E. Kounavis, K. Miki, J.B. Vicente, and D. Villela : "A Survey of Programmable Networks"; *ACM SIGCOMM Computer Communications Review,* (Apr. 1999).
2. Cristian, F. : "Understanding Fault-Tolerant Distributed Systems"; *Communications of the ACM,* Vol. 34 (2), (Feb. 1991) 56–78.

3. Floyd, S. and K. Fall : "Promoting the Use of End-to-End Congestion Control in the Internet"; *IEEE/ACM Transactions on Networking,* Vol. 7(4) (1999) 458–472.
4. Ghosh, A., M. Fry and J. Crowcroft : "An Architecture for Application Layer Routing"; Yasuda, H. (Ed), *Active Networks, LNCS 1942,* Springer:pp. 71–86. ISBN 3-540-41179-8 Springer-Verlag, (Oct. 2000).
5. The Network Simulator *ns2*, *http://www.isi.edu/nsnam/ns/*.
6. Nichols, K. and B. Carpenter : "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification"; Internet Draft, *http://www.ietf.org/internet-drafts/draft-ietf-diffserv-pdb-def-03.txt,* (Jan. 2001).
7. Saltzer, J.H., D.P. Reed, and D.D. Clark : "End-to-End Arguments in System Design"; *ACM Transactions of Computer Systems,* Vol. 2 (4), (Nov. 1984) 277–288.
8. Blake, S., D. Black, M. Carlson. E. Davies, Z. Wang, and W. Weiss : "An Architecture for Differentiated Services"; *http://www.ietf.org/rfc/rfc2475.txt*, (Dec. 1998).
9. Huston, G. : "Next Steps for the IP QoS Architecture"; *http://www.ietf.org/rfc/rfc2990.txt*, (Nov. 2000).
10. Goderis D., et al. : "Service Level Specification Semantics, Parameters and Negotiation Requirements"; *http://search.ietf.org/internet-drafts/draft-tequila-diffserv-sls-00.txt*, (Jul. 2000).