Evaluation of End-node Based Rate Allocation Schemes for Lambda Networks

Xinran (Ryan) Wu and Andrew A. Chien
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, 92093-0404, USA

Abstract

Dense Wavelength Division Multiplexing (DWDM), dedicated optical paths, high speed switches, and high speed routers are giving rise to networks with plentiful bandwidth in the core. In such networks, bottlenecks and congestion are concentrated at the edge and at end nodes.

We study how end nodes should efficiently and fairly manage capacity across multiple sessions with finite and unknown demands in such lambda networks. We show that it does not provide a sound solution by simply applying previous switch-based rate allocation schemes in ATM networks to the end nodes in lambda networks. We then compare two end-node based rate allocation schemes in such network environments. First, the end-node based distributed rate allocation algorithm [23] which is the heart of the Group Transport Protocol (GTP) [21, 22]. The algorithm allocates rates based on local information, and discovers the desired session rates. Second, a modified end-point only version of XCP [15] which moves the XCP router functions into end nodes (endpointXCP).

Using ns2 simulations, we compare GTP, endpointXCP, and two variants of TCP (New Reno and Highspeed TCP [8]) for high speed networks. We investigate convergence, efficiency, stability, and fairness behavior in lambda networks with high capacity, various round-trip times, and various traffic demands. Our results show that within the first 100 round-trip time, the TCP variants do not deliver the available network capacity. In addition, both GTP and endpointXCP deliver high throughput, endpointXCP fails to deliver the full max-min allocation to large users in some cases. In contrast, GTP supports high speed flows, leading to max-min fair allocation to all flows.

1 Introduction

Continuing advances in optical transmission are driving rapid increases in feasible network bandwidths at densities of over a terabit per optical fiber for both metro and long haul networks. A key driver is dense wavelength division multiplexing (DWDM) which allows each optical fiber to carry hundreds of lambdas (i.e. wavelengths) of 10 or 40 Gbps. These capabilities are being used to build high-speed traditional shared, routed internet networks, private dark fiber networks [17, 10], and based on new technologies for dynamic configuration, dynamic private lambda networks. Examples of these dynamic private networks include LambdaGrid [7] , OptIPuter [20], CANARIE [1], Netherlight [3], NAREGI [2], etc.).

Our research focuses in particular on networks where sets of high-speed private optical paths are constructed dynamically, forming a high-performance private network (e.g. the OptIPuter [20]). In such networks, the network bandwidth in the core of lambdas is high, matching or exceeding the speeds at which endpoints can source or sink network traffic. In short, the network of lambdas provides high bandwidth, very low loss packet communication. We call these high speed, private networks "lambda networks."

We make the following assumptions when modeling lambda networks.

1. With plentiful network bandwidth provided by private, dynamically configured lambdas, the network has no internal congestion. As a result, no detailed modeling of network internals is required. The rate control and allocation problem is reduced to efficient and fair sharing of source and sink capacities amongst the traffic sessions.

2. Each source and sink node has explicit knowledge of its capacity, usually determined by its network interface speed, local packet processing capacity or a shared link nearby. This information provides a budget for each node to allocate across its sessions.

3. Each session has a desired rate – the peak rate it can move data. The desired rate can be also interpreted as the maximum data handling speed of the applications using the network. The desired rate of each session is *unknown* to its source and sink, and can vary with time.

As end nodes become bottlenecks in lambda networks with plentiful bandwidth, we study how end nodes should manage their capacity across multiple sessions. Each end node needs to allocate its capacity based on incomplete information: local session status and approximations of session desired rates. This approach differs in two important ways from that addressed by session-based schemes (e.g., TCP and its variants [13, 8, 9, 24]). First, TCP and variants all manage single flows, providing rate and congestion control functionalities. They do not explicitly consider interactions amongst flows. Second, TCP and variants generally assume shared networks with internal congestion, so they focus on managing congestive packet loss within the network, not at the endpoints.

Achieving high-performance bulk data transfer has been a long standing research challenge for communication on high-bandwidth long-delay links. Other than session-based schemes described above, router or switch assisted approaches allocate router or switch capacity to its associated flows and feedback control signals based on traffic load or queueing delay. Such approaches include switch-based rate allocation schemes for ABR traffic in ATM networks [11, 6, 12] and router-assisted rate control schemes in IP networks (e.g., XCP [15]). We need to investigate whether such switch or router-based rate allocation schemes can be directly applied to perform end-node based rate allocation in lambda networks.

In this paper, we first formulate the end-node based rate allocation problem for lambda networks. We then compare three end-node based rate allocation approaches. First, schemes by applying to end nodes previous "flow marking" max-min fair rate allocation schemes for ABR traffic in ATM networks. Second, the end-node based distributed rate allocation algorithm [23] in Group Transport Protocol (GTP) [21, 22]. Third, a modified end-point only version of XCP [15] which moves the XCP router functions into the receiver (endpointXCP).

Our results suggest that previous switch-based rate allocation schemes for ABR traffic in ATM networks can not be directly applied as an effective solution. Ns2 simulations also show that both GTP and endpointXCP make more efficient utilization of end-node capacities than TCP alternatives. In addition, XCP leads flows converge quickly but fails to deliver full max-min allocation to large users in some cases. In comparison, GTP supports high speed flows to achieve max-min fair rate allocation among flows.

The remainder of the paper is organized as follows. In Section 2, we describe the end-node based rate allocation problem and present three different rate allocation approaches. In Section 3, we present evaluation experiment results, followed by a summary and discussion.

2 End-node Based Rate Allocation

2.1 The Rate Allocation Problem

Consider a lambda network $\mathcal{G}$ with a finite set $\mathcal{V}$ of nodes and a set $\mathcal{K}$ of K active sessions. Let set $\mathcal{D}$ denote the end-to-end delay of each active session. Let $\mathcal{V}^S$ and $\mathcal{V}^R$ denote the set of source and sink nodes, respectively. Note that multiple sessions may start or terminate at the same source or sink node. Consider a continuous-time model, and let each session $k \in \mathcal{K}$ be associated with a desired (peak) rate M_k. Let $x_k(t)$ be the instant rate of session k at time t. Let $\mathbf{x}(t) = (x_1(t), x_2(t), ..., x_K(t))$ be the rate vector of all active sessions at time t.

Let each source and sink $v \in \mathcal{V}$ have a capacity C^v. It is either the sending or receiving capacity of node v, when $v \in \mathcal{V}^S$ or $v \in \mathcal{V}^R$, respectively. Let $\mathcal{K}^v$ denote the set of sessions that node v participates in. Note that in our model each node v is either a sink or a source node, but not both.

Therefore $\mathcal{G}(\mathcal{V}, \mathcal{C}, \mathcal{K}, \mathcal{D}, \mathcal{M})$ characterizes an instance of the bandwidth sharing problem in lambda networks. The challenge of rate allocation and control in lambda networks is how to *efficiently* and *fairly* share the capacity of each source and sink among active sessions.

We describe the primary bandwidth sharing objectives as follows. First, the rate allocation (bandwidth sharing) algorithm should utilize the capacity of each source and sink efficiently while maintaining *feasibility*. Second, the rate allocation algorithm should treat all active sessions fairly. We adopt *Max-min Fairness* [5] as the criteria, as it is widely-used in wired, wireless and optical networks [14, 19, 18]. Max-min fairness maximizes the rates of the sessions with lower rates, and shares the remaining bandwidth until the network is fully utilized. Third, the distributed rate allocation algorithm should converge rapidly, reaching a unique rate allocation which is max-min fair. Fourth, the algorithm should be *adaptive* to flow dynamics. For instance, the rate allocation algorithm should avoid overflowing the sinks when session desired rates increase.

2.2 Max-min Rate Allocation in Packet Networks

The rate allocation problem in packet networks has been widely studied, especially in the context of allocating ATM switch capacity for ABR traffic. Charny et al. in [6] proposes the *Consistent Marking* scheme, which is proved to make session rates converge to the max-min fair rate allocation. Hou et al. in [12] studies the problem of generalized max-min rate allocation, which considers each session have constraints on minimum rate and peak rate. Such rate allocation schemes let each switch periodically calculate an advertised rate for each active sessions based on the switch

capacity, session current rates and minimum and peak rate constraints. Under such schemes each switch firstly identifies and marks the constrained sessions, which are likely bottlenecked by other switches, then set the advertised rate be the remaining bandwidth over the number of unmarked sessions.

These schemes can not be directly applied to solve the end-node based rate allocation problem in lambda networks. This is because in lambda networks the network switches are not bottlenecks and are not modelled. One way to extend such schemes is to move switch functions in such schemes to end nodes (sources and sinks). However, this is not a good solution for the problem we study for the following reasons:

First, previous works (e.g. [12]) assume explicit knowledge about session minimum and peak rate constraints. However, in the problem we study the session desired rate (peak rate) is unknown to end nodes and may change over time. This also suggests that any centralized or coordinated rate allocation scheme may not be a proper solution due to the lack of global knowledge.

Second, in most of the schemes described above each switch sends the same advertised rate to traffic sources. As a result, sessions with lower desired rate are allowed to send at a much higher rate immediately upon the increase of their demands. This will then severely overflow the sinks. We take a case of multipoint-to-point transfer for example to further illustrate this. Consider in a lambda network there are K ($K = 4$) sources and one sink, each with capacity 1. As shown in Figure 1, there is one active session between each source-sink pair. At equilibrium, the session desired rates, equilibrium rates and advertised rates by applying the distributed algorithm in [12] are shown in table 3. We see that each session gets the same advertised rate 0.7 from the sink. If at any later time the three sessions that have desired rate 0.1 increase their desired rate to 1, each of them will start sending at rate 0.7. This will result in an aggregate traffic of 2.8 at the sink, much higher than its capacity. This is especially a severe problem on long delay networks as it will take at least one round trip time for sinks to feedback a lower advertised rate in this situation. Therefore, when session desired rates are unknown, assigning the same rate control signal to all associated sessions may not work well in lambda networks. In the following two subsections we describe two schemes that provide sessions different explicit rate feedback.

2.3 Distributed Rate Allocation Algorithm in GTP

In [23] we describe the distributed algorithm in GTP, which only requires local node and actual session behavior to produce max-min fair rate allocation for arbitrary

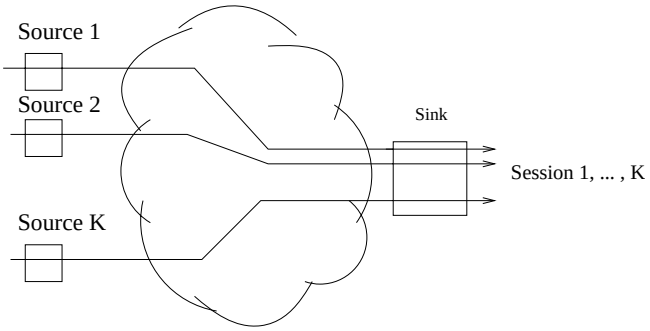

Figure 1. A single sink, multiple source topology.

Session	Session demand	Eq. rate	adv. rate
1	1	0.7	0.7
2	0.1	0.1	0.7
3	0.1	0.1	0.7
4	0.1	0.1	0.7

Table 1. 4-to-1 transfer

workloads. The proposed algorithm makes end-node based rate allocation and adaptation at each source and sink node. Specifically, in each control interval, each sink node allocates its capacity and sets expected rates $\hat{x}_k^r$ for each of its session k, using the status of all sessions terminating at the same sink:

$$\hat{\mathbf{x}}^r(t+1) = g(\mathbf{x}(t)),$$

where $g(\cdot)$ is the sink rate update function. The expected rate for each session k, $\hat{x}_k^r(t+1)$, at the sink is then fed back to the source. Each source node allocates its capacity to each session in similar fashion. Each source calculates the expected rate $\hat{x}_k^s$ for each of its sessions $k \in \mathcal{K}^v$:

$$\hat{\mathbf{x}}^s(t+1) = h(\mathbf{x}(t)),$$

where $h(\cdot)$ is the source rate adaptation function. As the network operates, the allowed rate for each session k is the minimum of $\hat{x}_k^s$ and $\hat{x}_k^r$. And the actual rate of session $k \in \mathcal{K}$ is then determined by the minimum of the expected source rate $\hat{x}_k^s$, the expected sink rate $\hat{x}_k^r$, and the session desired rate M_k, formally

$$x_k(t+1) = \min\{\hat{x}^s(t+1), \hat{x}^r(t+1), M_k\}. \quad (1)$$

We now take sink node for example to describe the rate allocation and adaptation scheme in detail. The idea is to adapt each session's current rate towards the max-min fair rate. In general, we try to maximize the session rates for sessions with lower desired rates by considering them earlier.

More specifically, in each control interval and at each iteration, the end node picks the session with the minimum rate among its undetermined sessions and assigns expected rates based on its current rate and the *target rate* x_f. Intuitively, the *target rate* approximates the max-min fair allocation of the remaining capacity over the undetermined sessions. If the current session rate is less than the target rate x_f, we adapt it toward the target rate, as

$$\hat{x}_k(t+1) = x_k(t) + \alpha(x_f - x_k(t)),$$

where $\alpha \in (0,1)$ is the adaptation step size ($0 < \alpha < 1$), and x_f is determined by the current remaining capacity over the number of undetermined sessions:

$$x_f = \frac{C^v - \sum_{i \in \mathcal{K}^v, \hat{x}_i(t+1) \neq 0} x_i(t),}{|\{j | \hat{x}_j(t+1) = 0, j \in \mathcal{K}^v\}|}. \quad (2)$$

We update x_f each iteration to reflect the changes in remaining capacity. When the minimum session rate of all undetermined sessions is greater than the *target rate* x_f, we distribute the remaining bandwidth evenly to the sessions, assigning each of them the same *target rate*, as

$$\hat{x}_k(t+1) = x_f, \text{ if } x_k(t) \leq x_f.$$

Under the scheme above, sessions with minimum rates are considered first, and their rates are given higher priority to continue increasing. It only stops increasing when it is restricted by its peer node or desired rate, or reaches the *target rate*. In the last case, the target rate is actually $C^v/|\mathcal{K}^v|$, a fair share of bandwidth when no sessions are limited by their desired rates or the peer nodes.

The source algorithm calculates source expected rates in fashion similar to the sink algorithm. Then the actual sending rate of a session can be calculated by Eq. 1.

Formal descriptions of the sink and source node algorithms, as well as formal proofs of its convergence and stability properties, can be found in [23].

2.4 endpointXCP

The Explicit Control Protocol (XCP) [15] is a novel transport protocol which decouples efficiency control from fairness control and makes each router allocate its capacity to traffic sources periodically. The efficiency controller in XCP uses MIMD to calculate the desired change of the aggregate traffic rate in a control interval, based on the persistent queue length and available bandwidth. MIMD helps to achieve fast convergence and high utilization. Then the fairness controller uses AIMD and "capacity shuffle" to compute the increase or decrease amount for each individual session, which guarantees constrained max-min fairness [16]. Compared with previous schemes on router/switch rate allocation, XCP gives different control feedbacks to various sessions, which is able to avoid the several overflow case described earlier.

Unmodified XCP is not directly applicable in lambda networks which often do not include routers, or much less XCP-enabled routers. To make a fairer comparison, we modify XCP, creating endpointXCP which moves the router functions in XCP to the sources and sinks. Each end node is aware of its capacity and all of the associated sessions, conduct the same rate allocation and adaptation and rate feedback defined for XCP router module. This allows us to investigate whether XCP-like scheme provides a good solution for the bandwidth sharing problem in lambda networks. To distinguish with the original XCP work, we call the modified scheme endpointXCP.

3 Comparison Studies

We conduct ns [4] simulations to compare GTP, endpointXCP, and two TCP variants, TCP NewReno and Highspeed TCP [8] with SACK1. We investigate their efficiency, convergence and stability, and fairness properties under various network configurations (multipoint-to-point and multipoint-to-multipoint) and different session demands (desired peak rates). The ns simulator version 2.28 is used, and all default parameter values of XCP remain unchanged in endpointXCP. Highspeed TCP is configured with $window\_option$ 8, and $max\_ssthresh$ is set to 50.

3.1 Multipoint-to-point Traffic

We first study a simple case where there are five sources and one sink node. The five sessions have different round-trip times varied from 10 ms to 90 ms. Each source and sink node's capacity is 500 Mbps. We firstly let each session have infinite desired rate, and the session trajectories under different protocols are depicted in Figure 2. In comparison, endpointXCP and GTP are able to achieve full utilization and fairness within 10 seconds, much higher than TCP variants. The endpointXCP has the fastest rate to converge.

We then consider the case where sessions have different desired rates. We let four of the five flows have the same low desired rate (25 Mbps) and 10 ms RTT. The other session's desired rate is 500 Mbps, and RTT is 50 ms. From Figure 3 We observe that only endpointXCP and GTP get high throughput in the first 10 seconds, and only GTP sessions achieve 100% link utilization. The fifth endpointXCP session can not fully utilize the remaining bandwidth.

To further illustrate this link utilization difference at the convergence state between GTP and endpointXCP, we construct a scenario in which we let one "fat" session share the same end node as various number of "thin" peer sessions. The aggregate desired rate from the "thin" peer sessions

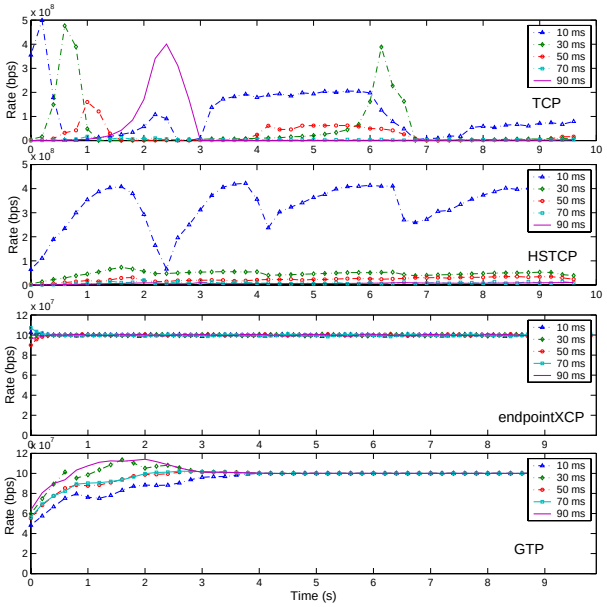

Figure 2. The 5-to-1 Case: Five sessions have different round-trip times: 10 ms, 30 ms, 50 ms, 70 ms, 90 ms. The sink capacity is 500 Mbps.

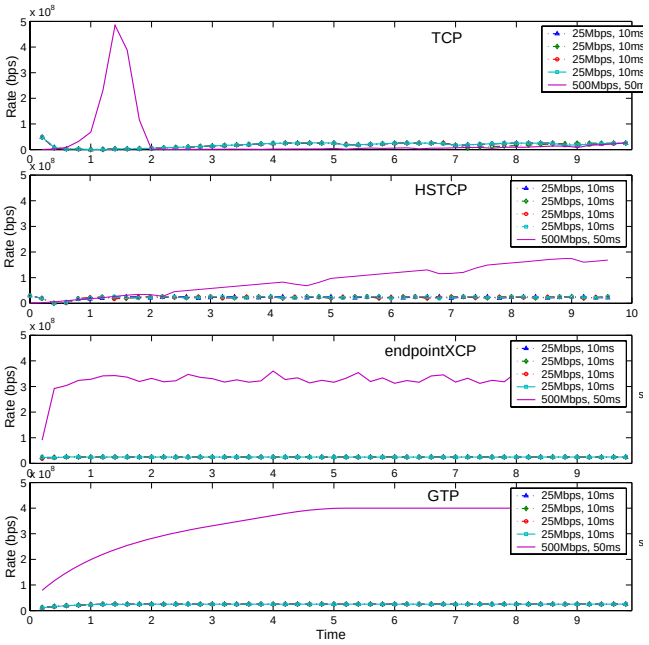

Figure 3. The 5-to-1 Case: Five sessions have different round-trip times: 10 ms, 10 ms, 10 ms, 10 ms, 50 ms. Four sessions have low desired (peak) rates of 25 Mbps. The sink capacity is 500Mbps.

is half of the sink capacity. Figure 4 shows the remaining link capacity utilization of the "fat" session, from which we see that the utilization ratio of the endpointXCP session declines when the number of "thin" peer sessions increases. This is due to the "bandwidth shuffle" effect of XCP. We refer to [16] for formal studies on the fairness constraints of XCP. In comparison, GTP is able to achieve optimized capacity utilization over various number of peer sessions.

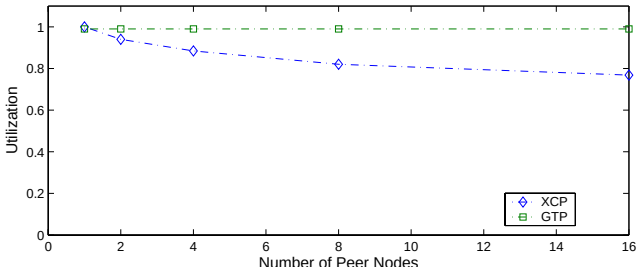

Figure 4. Comparison between endpointXCP and GTP: The link utilization of one "fat" session while sharing with different number of "thin" peer sessions.

3.2 Many-way Traffic (Network)

We evaluate the convergence properties of endpointXCP and GTP over a range of network sizes with traffic patterns. Now we show how endpointXCP and GTP behave in a 16 node lambda network. In such a lambda network, we let 8 nodes be traffic sources and 8 nodes be sinks. Each source node has 4 active sessions with randomly selected sink nodes, as shown in Table 2. The round-trip time is randomly generated between 1 ms and 100 ms, and the node capacity is 500 Mbps each. We assume infinite session desired rates in this case. Figure 5 shows the trajectories of endpointXCP and GTP for all 32 sessions. We see that both endpointXCP and GTP sessions converge to steady states. However as some of the endpointXCP sessions do not reach their optimized rates, the aggregated rate from endpointXCP sessions is 3.41 Gbps, compared with 3.50 Gbps from the max-min fair allocation as GTP achieves. Four highest sessions rates from endpointXCP and GTP are listed in table 3. This also shows that even though all the desired session rates are infinite in this case, endpointXCP may not achieve max-min fairness in certain complex networked scenarios.

To show the convergence properties of GTP for large networks, we define a distance metric between the current rate vector $\mathbf{x}(t)$ and the max-min fair equilibrium $\mathbf{x}^*$ (calculated for example by the global algorithm in [23]). Experiment-

Source	Sink	Source	Sink
1	1, 2, 2, 7	5	1, 3, 4, 5
2	1, 2, 3, 4	6	2, 2, 3, 4
3	2, 3, 4, 4	7	2, 5, 8, 8
4	3, 4, 4, 8	8	1, 4, 5, 6

Table 2. Connections between sources and sinks

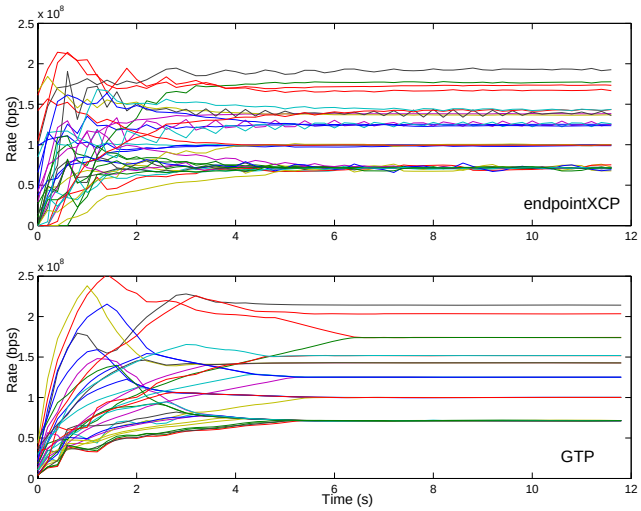

Figure 5. The trajectories of 32 sessions of endpointXCP and GTP in a randomly generated asynchronous 16-node network case. The RTT between each source and sink is randomly distributed between 1ms and 100ms.

ing with various distance metrics, we find one that works well is the 2-norm distance, defined as:

$$D(t) = \left[\sum_{i=1}^{K}(x_i(t) - x_i^*)^2\right]^{1/2}.$$

Therefore in the equilibrium state, we have $D = 0$. Figure 6 plots the 2-norm distances over time of the 30 GTP testing cases for randomly generated 128 node lambda-networks. In this case each node has the same link capacity which is normalized to 1. We see that although the initial distance varies due to different initial states, the GTP distributed algorithm causes the distance to decrease quickly, reaching equilibrium in 6 seconds.

Session	rate of XCP (Mbps)	rate of GTP (Mbps)
1	192.6	214.2
2	177.7	203.5
3	173.9	174.0
4	166.9	174.0

Table 3. Comparison between XCP and GTP: Four highest sessions rates

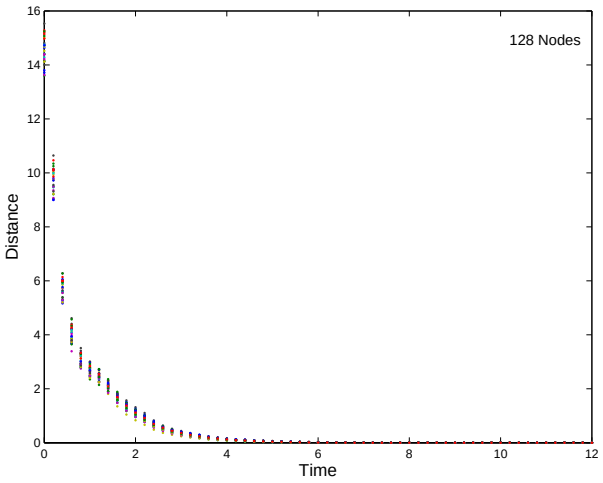

Figure 6. Network 2-norm distance for 30 test cases, 128 node network with 512 GTP sessions.

4 Conclusion

In lambda networks with plentiful bandwidth in the core, bottlenecks and congestion are concentrated at the edge and at end nodes. We formulated the rate control problem in lambda networks as how end nodes should asynchronously manage their capacity across multiple sessions with finite and unknown session demands. We compared three end-node based approaches: schemes by extending previous switch-based rate allocation schemes, the scheme that runs XCP control algorithm at end nodes, and the end-node based algorithm used in GTP. We showed that as session demands are unknown and may change over time, it is not proper to feedback the same rate control signal to all sessions. Simulation comparisons also showed that two end-node based approaches, endpointXCP and GTP, delivered more efficient and fair rate allocation than TCP variants in lambda network settings. In addition, XCP converged fast but failed to deliver full max-min allocation to large users in some cases. In comparison, GTP supported high speed flows by achieving max-min fair rate allocation among flows.

References

[1] Canarie. *http://www.canarie.ca*.

[2] Naregi. *http://www.naregi.org*.

[3] Netherlight. *http://www.netherlight.net*.

[4] Network simulator, ns version 2.28. *available at http://www.isi.edu/nsnam/ns/*.

[5] D. P. Bertsekas and R. Gallager. Data networks. *Prentice-Hall, Englewood-Cliffs, New Jersey, 1992*.

[6] A. Charny, D. D. Clark, and R. Jain. Congestion control with explicit rate indication. *Proc. IEEE International Conference on Communications (ICC'95)*, June 1995.

[7] T. DeFanti, C. Laat, J. Mambretti, K. Neggers, and B. Arnaud. Translight: a global-scale lambdagrid for e-science. *Communications of the ACM (CACM), 47(11), November 2003*.

[8] S. Floyd. Highspeed tcp for large congestion windows. *Internet draft*, August 2002.

[9] S. Floyd, S. Ratnasamy, and S. Shenker. Modifying tcp's congestion control for high speeds. *Available from URL http://www.icir.org/floyd/hstcp.html*.

[10] T. Greene. Dwdm is the right rx for new york presbyterian hospital. *Network World, 05/16/05, available at http://www.networkworld.com/news/2005/051605-presbyterian.html*.

[11] Y. Hou, B. Li, S. Panwar, and H.Tzeng. On network bandwidth allocation policies and feedback control algorithms for packet networks. *Computer Networks, vol.34, pp.481-501, 2000*.

[12] Y. T. Hou, H. Tzeng, S. S. Panwar, and V. P. Kumar. A generalized max-min rate allocation policy and its distributed implementation using the abr flow control mechanism. *Proceedings of IEEE INFOCOM 1998, pp.1366-1375, San Francisco, CA*, 1998.

[13] V. Jacobson. Congestion avoidance and control. *Proceedings of ACM SIGCOMM 1988 pp.314-329*, August 1988.

[14] P. Karbhari, E. Zegura, and M. Ammar. Multipoint-to-point session fairness in the internet. *Proceedings of IEEE INFOCOM 2003*.

[15] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for high bandwidth delay product network. *Proceedings of ACM SIGCOMM 2002, Pittsburgh, Aug 2002*.

[16] S. H. Low, L. L. H. Andrew, and B. P. Wydrowski. Understanding xcp: Equilibrium and fairness. *Proc. IEEE Infocom 2005, Miami, FL, March 2005*.

[17] C. Metz. The bright side of dark fiber optics. *PC Magazine, available at http://www.pcmag.com/article2/0,1759,1813381,00.asp*.

[18] B. Radunovic and J. L. Boudec. A unified framework for max-min and min-max fairness with applications. *Proceedings of 40th Annual Allerton Conference on Communication, Control, and Computing, Allerton, IL, October 2002*.

[19] B. Radunovic and J.-Y. L. Boudec. Rate performance objectives of multi-hop wireless. *Proc. IEEE Infocom 2005, Miami, FL, March 2005*.

[20] L. Smarr, A. A. Chien, T. DeFanti, J. Leigh, and P. Papadopoulos. The optiputer. *Communications of the ACM (CACM), 47(11), November 2003*.

[21] X. Wu and A. Chien. Gtp: Group transport protocol for lambda-grid. *Proceedings of CCGrid 2004*.

[22] X. Wu and A. A. Chien. Evaluation of rate based transport protocols for lambda-grids. *In Proceedings of the 12th IEEE International Symposium on High-Performance Distributed Computing (HPDC), Honolulu, Hawaii, June 2004*.

[23] X. Wu and A. A. Chien. A distributed algorithm for max-min fair bandwidth sharing. *UCSD Technical Report, available at http://csag.ucsd.edu*, 2005.

[24] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control for fast long-distance networks. *In proceedings of IEEE INFOCOM 2004, Hongkong, March 2004*.