Optimal Utility-based Internal Congestion Control in a Cluster-based Router

Qinghua Ye
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6H 2E8
qinghua@cs.ualberta.ca

Mike H. MacGregor
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6H 2E8
macg@cs.ualberta.ca

ABSTRACT

Optimal utility-based congestion control is an optimization approach to congestion control problems where the objective is to maximize the aggregate source utility over their transmission rates. It has been used to analyze Internet congestion control schemes. In this paper, we develop a discrete congestion control scheme to manage congestion internal to a cluster-based router. We simulate our proposed scheme with ns-3 and also evaluate this scheme in our cluster-based router prototype. Our results show that it is a very effective approach to improve the overall forwarding rate of the router by reducing the injection rate of traffic to the internal network when the router is under heavy load.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Store and forward networks*

General Terms

algorithms, measurement, performance

1. INTRODUCTION

Software routers based on commodity hardware and open-source operating systems are gaining more and more interest from both scientific researchers and business users. They have advantages in terms of price-performance and customizability compared to closed proprietary systems. For small and medium enterprises the price-performance advantage may be attractive. In a research setting, an open source router provides an opportunity to explore new algorithms and modify or extend router functions. However, in both contexts scalability and congestion control are of concern.

In order to address scalability we have proposed and evaluated an extensible and scalable cluster-based router framework[12]. Using this framework we have developed a prototype router based on a cluster of commodity processors interconnected by an InfiniBand network. Our cluster-based router performs very well and can be scaled as needed to deliver higher packet forwarding rates.

As in any distributed system, congestion control is a critical design issue. Congestion can affect both the forwarding rate and the latency of packets traveling through the router. If packets are sent too quickly from multiple ingress nodes to a single egress node, the egress node will become overloaded and its forwarding capability will be reduced. At this point, packets will be delayed and perhaps even dropped due to queue overflow at the egress. The congestion control scheme proposed here prevents the ingress nodes from overwhelming the egress nodes. It also reduces the waste of internal network bandwidth and CPU cycles by dropping packets as early as possible, so as to maximize the forwarding rate under overload. This is especially important given the correlation of components in commodity hardware[13].

Optimal utility-based congestion control is an optimization approach to congestion control problems where the objective is to maximize the aggregate source utility over their transmission rates. In this approach, the network links and traffic sources are viewed as processors of a distributed computation system to solve the maximization or optimization problem given the constraints of network link capacities. The traffic sources adjust transmission rates in order to maximize their own benefit, which is the utility minus bandwidth cost, while the network links adjust bandwidth prices according to their status to coordinate the sources' decisions on the evolution of transmission rates[10].

We apply this control approach to our cluster-based router, combined with backward explicit congestion notification within the router, from egress to ingress. We simulated the proposed scheme in ns-3 and evaluated in our cluster-based router prototype. Our measurements show an increase in forwarding rate of nearly 90% with a concomitant reduction in traffic in the internal network of up to 75%.

The rest of the paper is organized as follows. In section 2, we briefly introduce the architecture of our cluster-based router. In section 3, we describe how and where congestion occurs in our cluster-based router. The internal congestion control scheme is explained in section 4 and the simulation of this scheme is presented in section 5. We report the experimental evaluation in Section 6. We conclude our work in section 7.

2. CLUSTER-BASED ROUTER ARCHITECTURE

A cluster-based router [12] is composed of two main parts: the processing nodes and the interconnection network. In

Tech Report 2009. Department of Computing Science. University of Alberta. Canada. All right reserved.

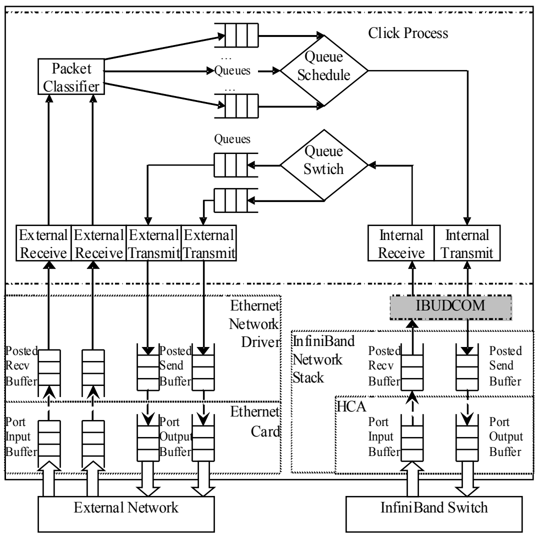

Figure 1: IP Forwarding Path

our prototype, the processing nodes are commodity server blades each equipped with one or more interface cards terminating the links to the outside world. The server blades also carry interfaces for terminating the links to the internal interconnection network. Our prototype uses 1 Gbps Ethernet links to the outside world. The internal interconnection network is a high throughput/low latency InfiniBand fabric. For more information regarding the cluster-based router, please refer to [12].

Fig. 1 gives the forwarding path in each processing node. In the following we will discuss ”ingress” and ”egress” nodes but it should be noted that these terms are relative to a given flow. There are no dedicated ingress or egress nodes as such - a particular processing node may function simultaneously as an ingress node for some flows, and an egress node for others.

An IP packet is received by the Ethernet driver at the ingress node where it will be captured by the external receive element in a Click kernel thread [6]. The packet then follows the processing path in the ingress node according to the defined Click configuration. At the end of this processing path, the internal interface passes the packet to the IBUDCOM (InfiniBand Unreliable Datagram COMmunication) layer and the packet traverses the internal network to the egress node. The packet is captured from the IBUDCOM layer by the internal interface at the egress node, switched to a specific outbound queue, and finally sent to the external network by the external interface through the Ethernet driver.

3. CONGESTION PROBLEM IN CLUSTER-BASED ROUTER

In any network, when the total demand for a resource is greater than the available capacity in any time interval, the resource is said to be congested for that time interval. The resources required in a cluster-based router include internal link bandwidth, switch buffer capacity, processor cycles, external transmission capacity, etc. [7]. Congestion control is a critical issue in the design of our cluster-based router. Congestion can affect both the forwarding rate of the router and the latency experienced by packets traveling through the router. The internal communication between processing nodes in our cluster-based router is based on the IBUDCOM service. This determines the potential congestion mechanisms on the path between the external interface on the ingress node and the external interface on the egress node.

Firstly, there is no packet dropping at the InfiniBand link layer. The internal interfaces on the cluster-based router nodes are called host connection adapters (HCAs). They connect each node to the internal InfiniBand switch. InfiniBand uses a link layer flow control scheme[3] to prevent an HCA from transmitting packets when the downstream HCA lacks sufficient buffer space to receive them. Specifically, InfiniBand manages the HCA oncard virtual link buffer rather than the host buffer. As a result no packets will be lost on the InfiniBand link layer and there can be no congestion between the HCA on the ingress node and the HCA on the egress node. It is important to recall that these HCAs face the internal network - they are not the external Ethernet interfaces.

Although the InfiniBand architecture guarantees no congestion between HCAs and thus across the internal switch, congestion can still arise inside the ingress node upstream of its HCA and inside the egress node downstream of its HCA. If the ingress node transmits packets at a rate that the InfiniBand link layer can service but faster than the egress node can process them, the egress node will eventually run out of posted receive buffers. At that point, packets will be discarded by the HCA on the egress node because it cannot place them in posted receive buffers for processing. Thus in order to avoid buffer overrun on the egress node, a congestion control scheme is required at or above the InfiniBand link layer.

Congestion can also arise in the queues on both the ingress and egress nodes. At an ingress node, incoming flows from multiple Ethernet interfaces must be queued for the single InfiniBand interface to the internal switch. As a result, the input queue to the HCA on the ingress node may overflow. At an egress node, due to the difference between the transmission capabilities of the external Ethernet interface and the reception capabilities of the InfiniBand HCA on that node, packets from multiple ingress nodes may be queued in the egress node or on the external Ethernet interface.

In conclusion there are two potential levels at which congestion may occur in our system: the Click packet processing layer and the IBUDCOM transport layer. In the first case, congestion can be detected by monitoring the queues on both ingress and egress nodes. The detection of congestion in the IBUDCOM transport layer is much more complex. Fortunately, we observed experimentally that congestion in the transport layer is always preceded by congestion in the egress queues if the scheduling weights are configured properly. In this case, before any packets are dropped by the transport layer due to the overrun of the egress InfiniBand receive buffer, the egress queues are already overloaded. The reason is that the receive buffer on the egress node is overrun only when the egress node processes received packets more slowly than the rate at which they arrive. If we give higher weights to receiving internal packets than to the transmission of external packets, then the receiving of internal packets will be scheduled more often than the transmission of external packets. As a result, if the receive buffer is overrun the egress queue is congested. Given this observation, we need to monitor only the status of the queues to detect potential congestion.

A variety of mechanisms might be considered to deal with congestion in these various locations. We start by considering the following: if the ingress nodes can be signalled to reduce their transmission rates to specific overloaded egress nodes then the drop rate of packets on those egress nodes will be reduced. This improves overall router throughput because it avoids wasting internal network bandwidth on dropped packets and reduces the proportion of cycles in both ingress and egress nodes which are spent on processing packets that will be dropped due to congestion. In this paper, we apply an optimal utility-based congestion control scheme.

4. OPTIMAL UTILITY-BASED CONTROL

Optimal utility-based congestion control has been applied in many ways. Based on the controlled objects, congestion control schemes can be divided into three classes - primal algorithms, dual algorithms and primal-dual algorithms[9]. In primal algorithms, users adapt their source rates dynamically based on route prices, and links select a static law to determine the link prices directly from the arrival rates at the links[5]. An example of a primal algorithm is TCP flow control[4]. In dual algorithms, on the other hand, links adapt the link prices dynamically based on the link loads, and users select a static law to determine the source rates directly from route prices and source parameters. REM[1] and AVQ[8] are good examples of dual algorithms. In primal-dual algorithms, both link prices and source rates are slowly adjusted so that they asymptotically converge to the optimal solution[9]. In today's Internet, the combination of TCP flow control algorithm and some active queue management algorithms like REM[1], RED[2] and AVQ[8] are examples of primal-dual algorithms.

In this section, we first present the network model of the internal network in the cluster-based router, and then develop the discrete congestion control scheme.

4.1 Internal Congestion Control As An Optimization Problem

Different from previous work, in the cluster-based router, instead of the links in a network, we focus on the state of the egress node. The model can be simplified to a network with only one egress node and can be described as follows.

Consider a network with unidirectional links. There is a finite forwarding capacity C associated with the destination. The destination is shared by a set S of sources, where source $s \in S$ is characterized by a utility function $U_s(x_s)$ that is concave increasing in its transmission rate x_s to the destination.

The goal of congestion control is to calculate the source rates that maximize the sum of the utilities $\sum_{s \in S} U_s(x_s)$ over x_s subject to capacity constraints. The congestion model can be written as:

$$P : \sum_{s \in S} U_s(x_s) \tag{1}$$

subject to

$$\sum_{s \in S} x_s \leq C \tag{2}$$

Equation (2) means that the aggregate source rate at the destination should not exceed the transmission capacity of the node. Since the utility function $U(x)$ is strictly concave and continuous, the optimal solution exists. Solving this problem centrally would require not only the knowledge of all utility functions, but also the knowledge of all the source rates and aggregate rate at destination, which makes it inflexible to changes in the numbers and/or types of sessions.

4.2 Decentralized Approach

From Equation (1), the objective function is separable in x_s. However, the source rates are coupled by the constraint (2). Fortunately, the dual theory of optimization leads us to a distributed and decentralized solution which applies the coordination of all sources implicitly[10].

As we described in the previous subsection, the utility function $U(x)$ is a differentiable concave function, so we can define the Lagrangian function of the original primal problem as:

$$\begin{aligned} L(x,p) &= \sum_{s \in S} U_s(x_s) - p(\sum_{s \in S} x_s - C) \\ &= \sum_{s \in S} U_s(x_s) - \sum_{s \in S} x_s * p + p * C \end{aligned} \tag{3}$$

The objective function of the dual problem is:

$$\begin{aligned} D(p) &= \max_{x_s} L(x,p) \\ &= \sum_{s \in S} \max(U_s(x_s) - x_s * p) + p * C \end{aligned} \tag{4}$$

and the dual problem is

$$D : \min_{p \geq 0} D(p) \tag{5}$$

In this dual problem, if p is interpreted as the price per unit bandwidth at the destination, then the first term of the objective function $D(p)$ is decomposed into S separable subproblems which can be solved at each source s with some information from the destination. Let

$$B_s(p) = \max(U_s(x_s) - x_s * p) \tag{6}$$

Hence, $x_s * p$ represents the bandwidth cost to source s when it transmits at rate x_s, and $B_s(p)$ represents the maximum benefit source s can achieve at the given price. Based on the congestion information p from the destination, each source s can solve the subproblem $B_s(p)$ without coordinating with other sources. Let $x_s(p)$ be the local optimizer to $B_s(p)$ at source s. Then from Equation (4), $x_s(p)$ is the local maximizer of the Lagrangian function (3).

In general, $x_s(p)$ may not be the primal optimal. However, according to the Karush-Kuhn-Tucker theorem, if there is minimizer p^* which is the optimizer of dual problem (5), and $x^* = x_s(p^*), s \in S$ is the maximizer of Equation (4), then (x^*, p^*) constitutes a saddle point for the function $L(x,p)$, and x^* solves the primal optimization problem P.

If we can solve the dual problem (5) to obtain p^* at the destination locally, then the original maximization problem P can be solved in a distributed manner using only the decentralized information available locally. In this case, p^* serves as a coordination signal that aligns individual optimality of (6) with social optimality of (1).

Based on the above theory, the congestion control problem can be generalized to tasks of finding distributed algorithms that can make sources to adapt transmission rates with respect to the destination price and make destinations to adapt prices with respect to loads.

According to the analysis, the optimal solution to the distributed congestion control problem satisfies:

$$\frac{\partial D(p)}{\partial x_s} = \frac{\partial U_s(x_s)}{\partial x_s} = U_s'(x_s) - p = 0 \tag{7}$$

$$\frac{\partial D(p)}{\partial p} = \sum_{s \in S} (-x_s) + C = 0 \tag{8}$$

These two equations can be solved in a distributed manner at the sources and destination. To reduce the extra overhead of transferring the link price, we only send the price from the destination to the sources at the beginning of each control interval, which results in a discrete control model.

Applying the gradient projection method where the source rate and destination price are adjusted according to the gradients (above two equations), we get the primal-dual algorithm. Here we set the utility function as proportional fairness utility function $U(x) = W * \log x$ [11].

$$\begin{aligned} x_s(k+1) &= [x_s(k) + k * x_s(k) * (U_s'(x_s(k)) - p(k))]_0^+ \\ &= [x_s(k) + k * (W - x_s(k) * p(k))]_0^+ \end{aligned} \tag{9}$$

$$p(k+1) = [p(k) + R * (\sum_{s \in S} x_s(k) - C)]_0^+ \tag{10}$$

Here

$$[g(x)]_y^+ = \{ \begin{array}{ll} g(x), & \text{y} > 0 \\ max(g(x), 0), & \text{y} = 0 \end{array}$$

4.3 Stability Analysis

Lyapunov's theorem gives a sufficient condition to check for stability of non-linear systems.

Lyapunov Theorem[11]: Consider a continuously differentiable functions $V(x)$ such that

$$V(x) > 0, \forall x \neq 0$$

and $V(0) = 0$.

1. If $\dot{V}(x) \leq 0 \ \forall x$, then the equilibrium point is stable.
2. In addition, if $\dot{V}(x) < 0, \forall x \neq 0$, then the equilibrium point is asymptotically stable.
3. In addition to the above two conditions, if V is radially unbounded, i.e.,

$$V(x) \to \infty, when \|x\| \to \infty,$$

then the equilibrium point is globally asymptotically stable.

Lyapunov's theorem can be used to prove stability of nonlinear systems, which makes it a good method to analyze system stability. However, the design of a Lyapunov function is not an easy task. Also, Lyapunov's theorem only provides sufficient conditions for stability. Even if we cannot find a Lyapunov function that satisfies the above theorem, we cannot conclude that the system is unstable.

For discrete systems, i.e. $x(k+1) = f(x(k))$, then we can use $V(x(k))$ and $\Delta V(x(k))$ instead of $V(x)$ and $\dot{V}(x)$. As long as $V(x(k))$ and $\Delta V(x(k))$ satisfy the conditions of Lyapunov's theorem, then the same conclusions regarding stability stand.

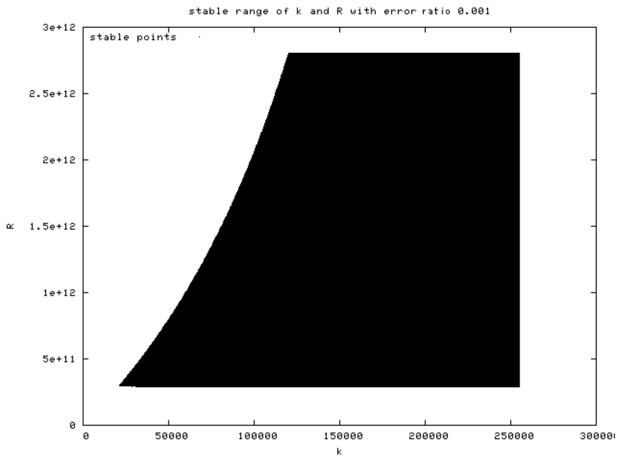

Figure 2: Stability Region - by Simulation

In [9] Srikant simulates a real system with a continuous model and designs a Lyapunov function for the primal-dual algorithms. This leads to a proof that the primal-dual algorithm is stable as long as $U(x)$ is strictly concave on x and $k(x) > 0$ for any $x > 0$ is a non-decreasing continuous function.

Similarly, in our optimal utility-based internal congestion control, if we use the continuous model

$$\dot{x_s} = [k * (W - x_s * p)]_x^+$$

$$\dot{p} = [R * (\sum_{s \in S} x_s - C)]_p^+$$

and set

$$V(x, p) = \sum_{s \in S} \int_{x_s^*}^{x_s} \frac{u - x_s^*}{k * u} du + \int_{p^*}^{p} \frac{v - p^*}{R} dv$$

then our optimal utility-based internal congestion control scheme is stable since $w * log(x)$ is a strictly concave function on x and $K(x) = k * x > 0$ for any $x > 0$ is non-decreasing continuous function (See appendix for proof).

However, in a discrete system, the stability condition is different from the continuous case. We demonstrate this via numeric simulation by running our discrete model and plotting the phase space of stable points in Fig.2. In this simulation, we set the maximum bandwidth C to be 764500 packets per second. We also set the allowed error ratio to 0.001 which means that the system with certain k and R is said to be stable if the source rates converge to the optimal allocated bandwidth with error ratio less than 0.1%. For the discrete system, the stable region starts at around $k = 25000$ and $R = 3 * 10^{11}$. Note that we only simulate with $k < 255000$ and $R < 2.8 * 10^{12}$. Whereas the continuous system is stable for any k and R, the discrete system is only stable for values of k and R in the black region of the phase plot.

5. SIMULATION WITH NS-3

To study the behavior of our congestion control algorithm, we simulated it using the ns-3 network simulator. ns-3 is a discrete-event network simulator that is targeted primarily for research and education use. The network components in ns-3 are easily implemented and extended.

ns-3 implements most key network components, from network devices to socket APIs and applications. However, it limits itself to the standard network stack or Internet protocol suites. To simulate novel network stacks, extra work is necessary to extend existing network components.

In terms of router components, ns-3 implements the standard IP forwarding path in the Ipv4L3Protocol class. To simulate the internal congestion control scheme in the cluster-based router, we extended the Ipv4L3Protocol class to include the packet classifier, queue management and scheduling, and backward explicit congestion notification (BECN) functionalities. With the extended Ipv4L3Protocol instances installed in the ns-3 nodes, we can simulate the behaviors of processing nodes in the cluster-based router and study the effects of different congestion control schemes in the cluster-based router.

In this section, we first introduce the IP forwarding path implemented in our simulation, then we describe the simulation setup, and lastly we present and analyze the simulation results.

5.1 IP Forwarding Path

In ns-3, the IP forwarding path is implemented in a class called Ipv4L3Protocol. To make use of an Ipv4L3Protocol object, network devices should be attached to a node configured with an Ipv4L3Protocol instance. The Ipv4L3Protocol class provides the interface to add network interfaces with given network devices. The Ipv4L3Protocol instance registers the Receive callback method with the network devices. In this way, any IP packet received by the network devices will be forwarded to the Ipv4L3Protocol instance by the Receive callback method. Once the Ipv4L3Protocol instance gets the packet, the packet will go through the IP forwarding path as implemented.

The cluster-based router is composed of several routing nodes which we called processing nodes which can be ingress or egress nodes in terms of an individual flow. However, a network interface can only be an external interface or an internal interface. By external interface we mean a network interface that connects to the external network, while by internal interface we mean a network interface that connects to the internal network which connects all the processing nodes together. Therefore, we extended the Ipv4L3Protocol class implemented in ns-3 to distinguish these two different kinds of network interfaces. A method called SetInternalDevice sets a network device to be internal. Otherwise, the network device is considered as external by default. Once a network device is marked as internal, its Ipv4L3Protocol instance will register with it to receive BECN type packets in addition to the general IP packets.

Figure 3 describes the IP forwarding path in our simulation. Once a packet is received in the network device, the Receive method is called by the network device. According to the network device from which the packet is received, we can classify the packet to be an external traffic packet or an internal traffic packet.

If the packet belongs to the external traffic flow, after being looked up in the routing table with the destination IP, the packet will be pushed to a classifier if the next hop of the packet is another processing node in the cluster-based router. There it is switched to different queues according to the preset classification rules. Packets in these queues will be scheduled to be sent from the internal network interface. The scheduler controls the pace of putting packets into the output queue of the internal network device. If the output queue is available, the scheduler schedules a packet from upstream queues and puts the packet into the output queue. Whenever the internal device is not busy, a packet will be dequeued from the output queue for transmission. A packet being dequeued for transmission from the output queue will not be dropped by the network device since the packet is forwarded to the internal network device only when the internal network device is not busy. Without the output queue in the original implementation of Ipv4L3Protocol and without checking the status of the network device, a packet forwarded to the network device may be dropped due to the network device being busy.

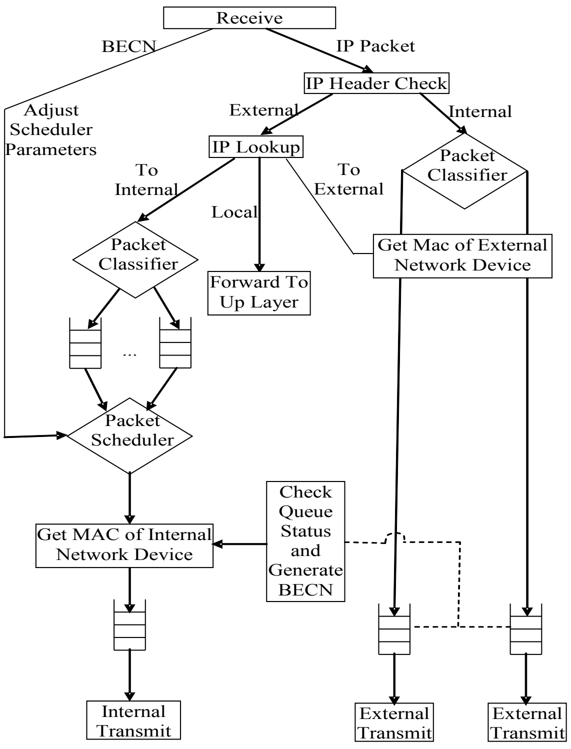

Figure 3: IP Forwarding Path in Simulation

If a packet belongs to the internal traffic flow which is received by the internal network device, or a packet belongs to the external traffic flow but will be transmitted out through the same processing node, the packet is switched to the output queue of the external network device where it will be transmitted out to the external network. A global monitoring process is scheduled periodically to check the status of the output queues of external network devices and send BECN packets to the other processing nodes if necessary.

If a BECN packet (only from internal network device) is received by a processing node, the parameters in the scheduler are adjusted, which updates the rate of packets being scheduled from each queue.

5.2 Simulation

Having all the functional components of a processing node in the cluster-based router ready, we can set up the simulator to simulate the internal congestion control scheme in the cluster-based router.

5.2.1 *Cluster-based Router*

First of all, we set up the cluster-based router. We extended the Ipv4L3Protocol class in ns-3 to include all of the basic functional components of processing nodes in the cluster-based router. We configured the nodes installed with the extended Ipv4L3Protocol instance to make them behave like processing nodes. Then we set up a CSMA link with 10G data rate and 1ns delay to connect all the processing nodes together to simulate the InfiniBand internal network. The network interfaces of this CSMA link on the processing nodes are configured to be internal interfaces.

Simulating the InfiniBand internal network of our real prototype with a CSMA link is an approximate, but very useful choice. Firstly, due to the capacity difference between the external network and internal network, the internal network will never be a bottleneck in our simulation. Secondly, in the real system, there are statistical factors which can not be captured by simulating a deterministic switch fabric, and we have not as yet characterized the Infiniband fabric. The collision detection and exponential backoff algorithms implemented in CSMA bring a statistical flavor to the behavior of the fabric that makes the simulation more realistic.

The processing nodes also connect to the external network via Ethernet interfaces. We configured one external interface for each processing node.

After setting up the network interfaces and devices, we configured the extended components in the Ipv4L3protocol instance: we initialize the queues for each class of traffic, configure the classification rules, set the global time interval to schedule packets from output queues of network devices for transmission, set the congestion control interval, set the congestion control scheme and corresponding control parameters, and set the nodes to be processing nodes.

5.2.2 *Simulation Environment*

Now we have a cluster-based router ready for processing packets. To study its behavior, traffic sources and destinations are needed. The goal of this simulation is to study the behavior of congestion control, so we configured three traffic sources and one traffic destination.

Four general nodes installed with TCP/IP network stack are created and connected to the cluster-based router processing nodes, with one general ns-3 node connecting to one processing node. On three of them, we install ns-3 on/off traffic generator applications to generate traffic flow with specified traffic pattern, and the other node is installed with ns-3 traffic sink application.

The patterns of the traffic flows generated on the three traffic source nodes are configured with the traffic sink node as destination. In this setup, three processing nodes of the cluster-based router receive packets from external networks and forward the packets to the other processing node, which forwards the packets to the traffic sink node in the external network. By configuring the traffic rates and patterns on the traffic sources, we are able to simulate different congestion states and study the behaviors of different congestion control schemes.

Since small packets stress the router more than large packets at the same aggregate bit rate because more packets have to be handled per second, we use 64B packets in our simulation. Previously [13], we measured the maximum transmission rate of the Intel E1000 Ethernet network port on PCI-X64/100 bus as 764500 packets/second for 64B packets. So we set the capacity of the transmission rate on the destination node to this value.

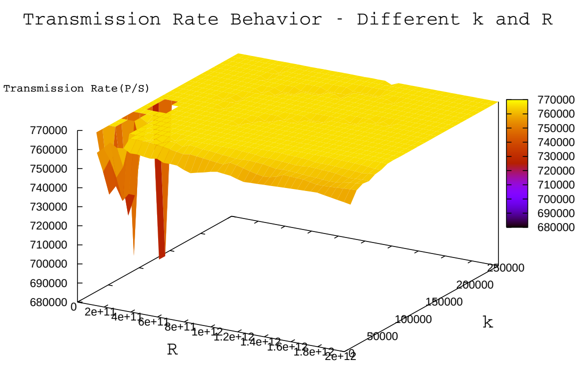

Figure 4: Transmission Rate on the Egress Node

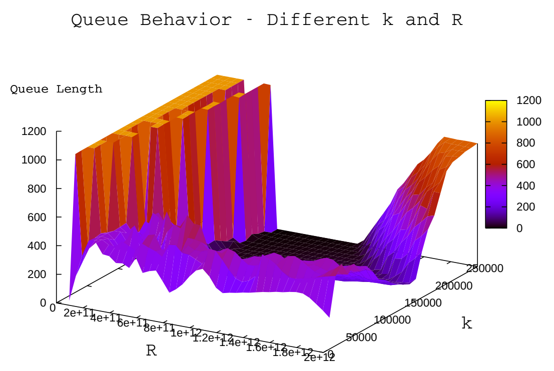

Figure 5: Average Output Queue Length

5.2.3 *Simulation Data Collection*

To study different congestion control schemes including the throughput and convergence of the system, detailed system state information is required to facilitate the analysis process. In the implementation of the processing nodes, we added many status logging plugins to collect the status of the system during simulation. These include queue length, receiving and transmitting rate of network devices, and source distribution of received and transmitted packets.

5.3 Simulation Results and Analysis

In our simulation, we run a set of simulations given different k and R, which are factors used to calculate the proportional fairness utility and the congestion price. We measure the average transmission rate of the external network interface and the average length of the external queue which is used to buffer the packets that will be transmitted from the external network interface. In our simulation, the ingress nodes are all offered the same amount of traffic unless otherwise noted.

5.3.1 *Stability Phase Space*

As shown by the plot in Fig.2, the proposed algorithm is stable only in some specific region, although the continuous

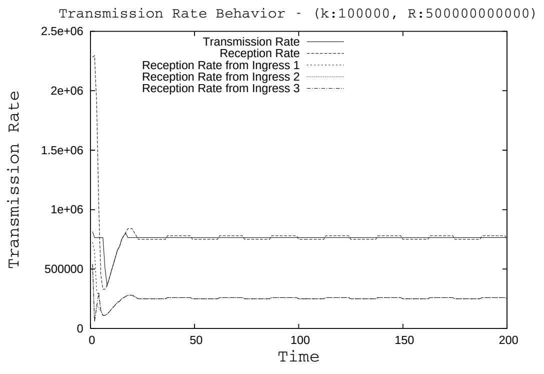

Figure 6: Transmission Rate Convergence

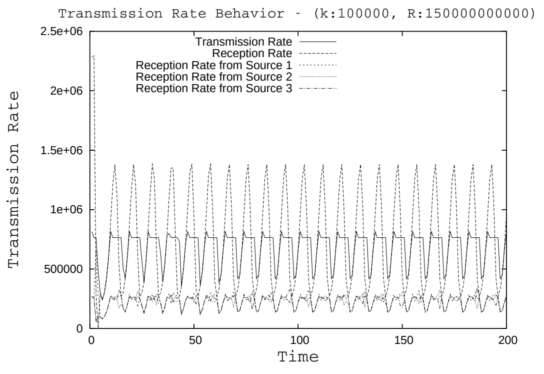

Figure 7: Transmission Rate Convergence

model can be proved to be stable for any given k and R. By "stable" we mean the transmission rate on the destination and the sum of source transmission rates are close to each other and are also close to the destination transmission capacity, and the transmission output queue length on the destination node is relatively low. Due to the discrete control accuracy on the source transmission rates and the price calculation on the destination, a small variance of the transmission rates and the queue length should be acceptable.

Fig.4 shows average transmission rates on the destination node given different k and R after 500 iterations. We can see that for most values of k and R, the transmission rate is close to the transmission capacity of the destination node.

Fig.5 shows the average output queue length on the destination node. From this figure, we can see that, for a certain range of k and R (when $R > 3*10^{11}$ and $k > 50000$), the output queue size remains small, which means that the reception rate on the destination node is not exceeding the transmission capacity. Combined with the fact that the transmission rate is close to the transmission capacity (Fig.4), we can conclude that the reception rate and the transmission rate converge to the transmission rate and the control scheme is stable while k and R are in this range.

5.3.2 Convergence given different k and R

Fig.4 and Fig.5 give the range of k and R over which the proposed internal congestion control is likely to be convergent. Now we study the behavior of our internal congestion control with given k and R. Picking $k = 100000$ and $R = 5 * 10^{11}$, where the average output queue length is small, we plot the behavior of the internal congestion control in Fig.6 when the three sources offer traffic at the same rates. We collect the reception and transmission rates of the destination node, and the transmission rates of the source nodes at an interval which equals the control interval. We can see that after 20 intervals, the rates start to be convergent and the reception and transmission rates of the destinations nodes stabilize near the transmission capacity of the destination node.

Given different values for R with the same k, the congestion control scheme behaves differently. With smaller R, the response time becomes shorter and the control step becomes bigger, which may result in the system not converging, or

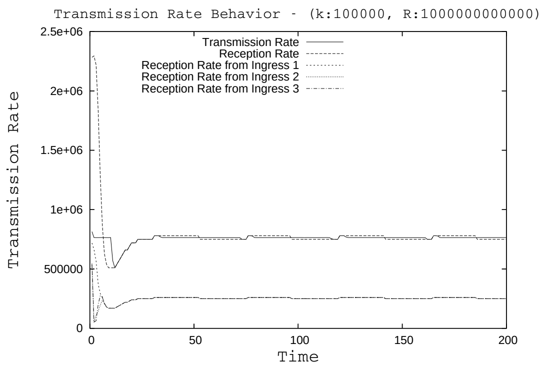

Figure 8: Transmission Rate Convergence

worse yet becoming unstable. Fig.7 plots one unstable case with $R = 1.5 * 10^{11}$. With larger R, the response time becomes longer and the control step becomes smaller, and the system reacts slowly. Fig.8 gives the behavior of our control scheme at $R = 10^{12}$.

Given different values for k with the same R, the congestion control scheme again behaves differently. With smaller k, the response time is longer while the response time is shorter with bigger k. Fig.9 and Fig.10 give the behaviors of our congestion control scheme with $k = 20000$ and $k = 150000$ when R is set to $5 * 10^{11}$.

5.3.3 Fairness

In the above figures all sources offer traffic at the same rates and all sources are serviced at the same pace. So the proposed congestion control scheme is fair to all sources. Given varied source traffic rates, we find that the congestion control is also stable and fair to all sources. Fig.11 presents the behavior of our control scheme with $k = 100000, R = 5 * 10^{11}$ when source 1 is given 500Mbits/second traffic, source 2 is given 50Mbits/second traffic, and source 3 is given 5Mbits/second traffic. We can see that source 3 is 100% satisfied since it requests less than 1/3 of bandwidth and source 2 and 3 are given same bandwidth since they

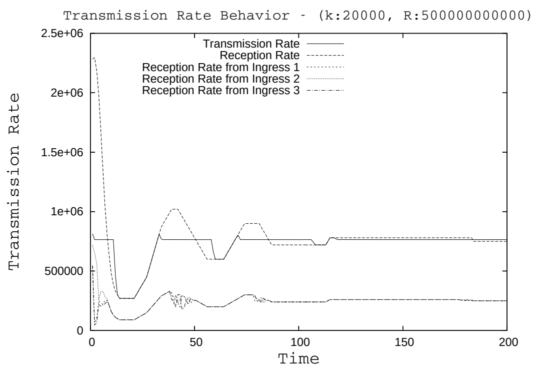

Figure 9: Transmission Rate Convergence

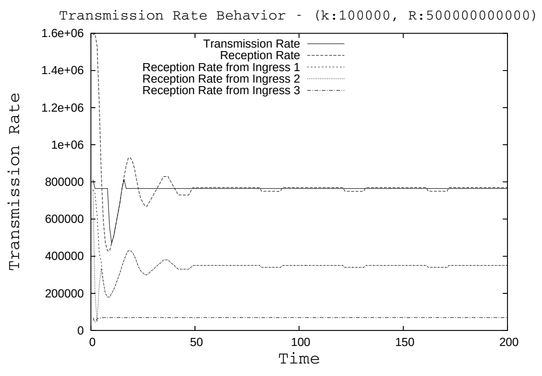

Figure 11: Faireness Given Varied Source Traffic

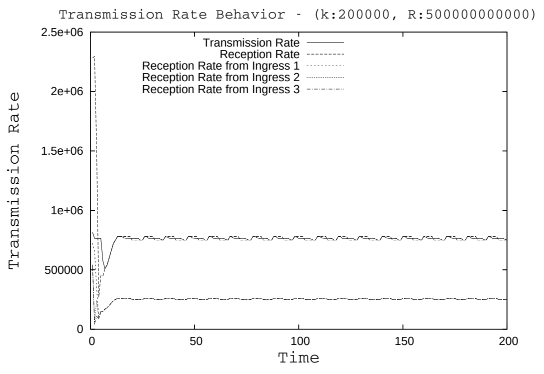

Figure 10: Transmission Rate Convergence

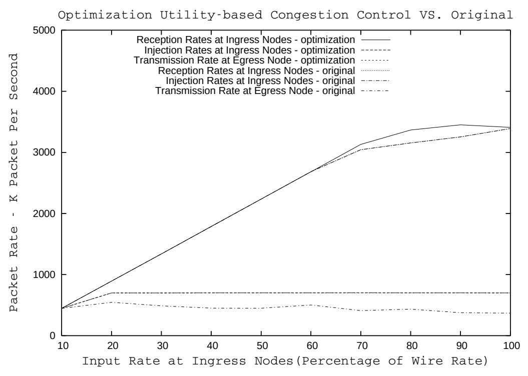

Figure 12: Optimal Utility-based Congestion Control vs. Original

both request more than the system can service (overlapped in Fig.11).

6. EVALUATIONS IN THE REAL SYSTEM

In this section, we evaluate our optimal utility-based congestion control scheme on our cluster-based router prototype. We implemented the scheme as described in Fig.3. and in Section 5.2.

6.1 Experiment

Our prototype router is composed of a cluster of four SunFire X4100 nodes interconnected by a Mellanox InfiniBand switch. Each SunFire node has two AMD Opteron 254 processors operating at 2.8GHz, 4 GB of registered DDR-400 SDRAM, four Intel E1000 Ethernet ports connected to a 100MHz PCI-X bus, and one Voltaire InfiniBand 4x PCI-X HCA installed in a 133MHz 64-bit PCI-X slot. A Mellanox MTS2400 24-port Modular InfiniBand Switch System that can support 10 Gbps switching rate at each port is connected to the HCA on each node. Packets are generated by a commercial traffic analyzer and sent to the router external Ethernet ports for forwarding. Packets that are successfully forwarded through the router return to the traffic analyzer via the router Ethernet ports.

In order to evaluate the effectiveness of our proposed congestion control scheme we compare the behaviors of the cluster-based router first with, and then without, internal congestion control. The traffic analyzer (with four ports) is configured to offer Ethernet traffic to three of the nodes (one port each) with the destination being a subnet reached via the fourth node. We increase the offered load until the egress port is heavily overloaded - the offered load is over 300% more than the maximum load the egress port can forward. As well as acting as a traffic source and sink, the network analyzer is used to monitor the rate at which Ethernet packets are sent to each ingress port and the rate at which the egress port returns packets to the traffic analyzer. As in the simulation we use 64B packets in our tests. Also different from the simulation, due to the correlation between different components on the PCI bus [13], we set the transmission capacity of the egress port to 700K packets/second which is less than the transmission capacity of the port (760K packets/second) measured when there is no other traffic on the system.

6.2 Measurements

We first compare the transmission rates on the egress

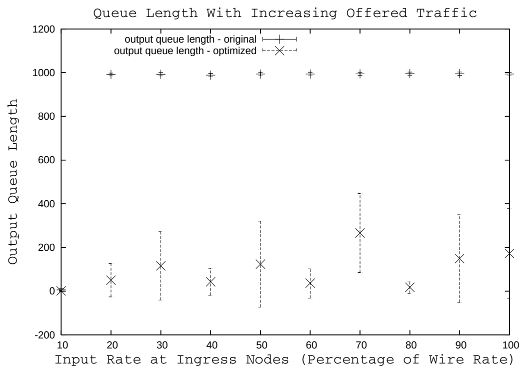

Figure 13: Optimal Utility-based Congestion Control vs. Original

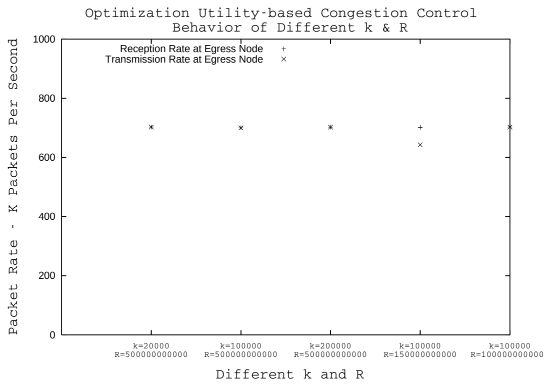

Figure 15: Behavior of Optimal Utility-based Congestion Control With Different k&R

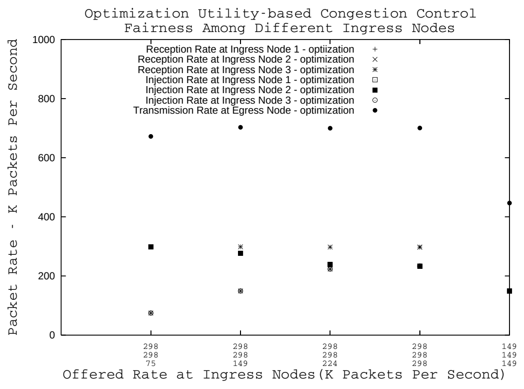

Figure 14: Fairness in Optimal Utility-based Congestion Control

node with and without the congestion control scheme applied when all the ingress nodes are fed at the same rates. As plotted in Fig.12, in the original situation, with increasing offered traffic on the ingress nodes, the ingress nodes inject more traffic into the internal network, which causes the transmission rate on the egress nodes to decrease after the offered load exceeds the point of "congestion collapse". With our congestion control scheme, the ingress nodes only inject the packets which can be processed on the egress node to avoid overwhelming the internal network, the internal network port and the external port on the egress node, which results in a stable transmission rate on the egress node. The injection rate overlaps with the egress transmission rate line.

Fig. 13 presents the average queue length and standard deviations of queue length on the egress node with and without our congestion control scheme. It shows that our proposed scheme is effective in reducing the congestion on the egress node which helps to keep the output queue small.

We also plot the fairness between ingress nodes when they are offered different traffic loads in the case that the congestion control scheme is in place. Fig. 14 gives the packet reception and injection rates at each ingress node as a function of offered load. If all ingress nodes are offered the same load, then they inject packets into the internal network at the same rate and get the same share of the transmission on the egress node. If one ingress node is offered less load than the other two, the more heavily loaded nodes are punished more than the lighter loaded node and all overloaded ingress nodes still get an equal share until the more lightly loaded source is 100% serviced.

The different behaviors of our congestion scheme with different k and R are also presented in Fig.15. We measure the average transmission rates of the egress port when the ingress nodes are offered traffic at the same rates. These results are for the case where our congestion control scheme applied. For most settings (as shown in Fig.6,8,9,10), the transmission rate is close to the set transmission capacity of the egress port, while in the case shown in Fig.7, the transmission rate is much less. Note that for all 5 cases, the injection rates are same as the set transmission capacity of the egress port. The larger fluctuation of the injection rate as shown in Fig.7 causes overflow of the egress queue, which results in packet drops and low average transmission rate on the egress nodes. For all other cases, because the fluctuation of the injection rate is small and the set transmission rate of the egress port is less than the actual transmission capacity, the fluctuation is absorbed by the difference between the set and actual transmission capacity, which results in less dropping of packets.

Given the above real system experiments and the ns-3 simulations presented in section 5.3, we can conclude that our discrete optimal utility-based internal congestion control model is effective with certain range of parameters, and the ns-3 simulations can be used to study the behavior of the scheme and predict the range of parameters which make the control scheme efficient and the system stable.

7. CONCLUSION

Congestion control is a universal problem in distributed systems. The specific characteristics of the forwarding path through our cluster-based router make the detailed features of the congestion problem unique. In particular, the link-layer flow control implemented as part of the InfiniBand standard pushes congestion outward from the internal network to the dependent ingress and egress nodes.

In this paper, we propose a congestion control scheme which uses BECN notification internal to the router along with optimal utility-based control scheme. We prove the stability of the continuous model of the proposed congestion control scheme and also check the stability of the model via simulation of the discrete system. The application of the scheme to our cluster-based router prototype shows that the stability of the real system follows the results predicted by the simulation.

Both the simulation and experimental evaluation of the scheme show that our proposed congestion control scheme is effective, efficient and fair if we set the parameters to keep the router in its stable range. The control scheme is effective in preventing offered traffic from causing congestion collapse, achieves efficiency by reducing the waste of internal network bandwidth and CPU cycles by dropping overloaded packets early, and allocates throughput fairly among different sources. With this congestion control scheme in place, the forwarding rate of our router is increased by up to 90% under overload while the traffic in the internal network is reduced by up to 75%.

8. REFERENCES

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: active queue management. *IEEE Network*, 15:48–53, 2001.

[2] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[3] IBTA. *InfiniBand Architecture Specification.* InfiniBand Trade Association.

[4] V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988. ACM.

[5] F. P. kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[6] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[7] A. Kumar, A. Maniar, and A. S. Elmaghraby. A fair backward explicit congestion control scheme for ATM network. In *ISCC '99: Proceedings of the The Fourth IEEE Symposium on Computers and Communications*, pages 452–457, Washington, DC, USA, 1999. IEEE Computer Society.

[8] S. S. Kunniyur and R. Srikant. An adaptive virtual queue (AVQ) algorithm for active queue management. *IEEE/ACM Transactions on Networking*, 12:286–299, 2004.

[9] S. Liu, T. Basar, and R. Srikant. Controlling the Internet: a survey and some new results. In *Proceedings of the 43nd IEEE Conference on Decision and Control*, pages 3048–3057, Maui, Hawaii USA, December 2003.

[10] S. H. Low and D. E. Lapsley. Optimization flow control - I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[11] R.Srikant. *The Mathematics of Internet Congestion Control.* Birkhauser, 2003.

[12] Q. Ye and M. H. MacGregor. Cluster-based IP router: Implementation and evaluation. In *IEEE International Conference on Cluster Computing*, pages 1–10, Barcelona, Spain, September 2006.

[13] Q. Ye and M. H. MacGregor. Hardware bottleneck evaluation and analysis of a software PC-based router. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systerms*, pages 480–487, Edinburgh, UK, June 2008.

9. APPENDIX

$$V(x,p) = \sum_{s\in S}\int_{x_s^*}^{x_s} \frac{u - x_s^*}{k * u}du + \int_{p^*}^{p} \frac{v - p^*}{R}dv$$

is a qualified Lyanpunov function so that the internal congestion control algorithm

$$\dot{x_s} = [k * (W - x_s * p)]_x^+$$

$$\dot{p} = [R * (\sum_{s\in S} x_s - C)]_p^+$$

is globally, asymptotically stable.

Proof:

$$\begin{aligned}
\dot{V} &= \sum_{s\in S} \frac{x_s - x_s^*}{k * x_s} * \dot{x_s} + \frac{p - p^*}{R} * \dot{p} \\
&\le \sum_{s\in S} \frac{x_s - x_s^*}{k * x_s} * k(w - x_s * p) + \frac{p - p^*}{R} * R(\sum_{s\in S} x_s - C) \\
&= \sum_{s\in S} \frac{x_s - x_s^*}{x_s} * (w - x_s * p) + (p - p^*) * (\sum_{s\in S} x_s - C) \\
&= \sum_{s\in S} \frac{x_s - x_s^*}{x_s} * (w - x_s * (p - p^* + p^*) \\
&+ (p - p^*) * (\sum_{s\in S} x_s - x_s^* + x_s^* - C) \\
&= \sum_{s\in S} (-x_s + x_s^*) * (p - p^*) + (p - p^*) * \sum_{s\in S} (x_s - x_s^*) \\
&+ \sum_{s\in S} \frac{x_s - x_s^*}{x_s} * (w - x_s * p^*) + (p - p^*) * (\sum_{s\in S} x_s^* - C) \\
&= \sum_{s\in S} \frac{x_s - x_s^*}{x_s} * (w - x_s * p^*) + (p - p^*) * (\sum_{s\in S} x_s^* - C) \\
&\le 0
\end{aligned} \tag{11}$$

Since $\sum_{s\in S} \frac{x_s - x_s^*}{x_s} * (w - x_s * p^*) = 0$ when $x_r = x_r^*$, and $\frac{w}{x_r}$ decreases as x_r increases, $\sum_{s\in S} \frac{x_s - x_s^*}{x_s} * (w - x_s * p^*) \le 0$. On the other hand, $\sum_{s\in S} x_s^* \le C$and $p^* = 0if \sum_{s\in S} x_s^* < C$, so $(p - p^*) * (\sum_{s\in S} x_s^* - C) \le 0$.