International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:7, 2009

QoS Routing in Wired Sensor Networks with Partial Updates

Arijit Ghosh and Tony Gigargis
Center for Embedded Computer Systems
School of Information and Computer Science
University of California
Irvine, CA 92697
Email: {Arijit.Ghosh,givargis}@uci.edu

***Abstract*—QoS routing is an important component of Traffic Engineering in networks that provide QoS guarantees. QoS routing is dependent on the link state information which is typically flooded across the network. This affects both the quality of the routing and the utilization of the network resources. In this paper, we examine establishing QoS routes with partial state updates in wired sensor networks.**

I. Introduction

Ethernet and ATM-based video surveillance network (VSN) using intelligent cameras connected to PC-class devices [1][2] is an example of a popular and practical resource-rich sensor network. Such resource intensive applications have strict specifications on delay, delay jitter (or delay variation) and loss guarantees. It is therefore very important to design a network that provides end-to-end QoS guarantees. A sensor network application can be abstracted in terms of a data flow computational model. A flow in this model corresponds to a stream in that it is either potentially unbounded or, more typically, has a long life. While an application can be architected in many different ways, we are interested in a distributed architecture where the source and destination of a flow are mapped on to different network nodes. Thus without loss of generality, we can state that the problem of providing network QoS reduces to the problem of providing QoS for point-to-point per-application flow.

The delay of a flow is affected by many aspects of the network, for example queueing discipline and packet scheduling. However, in this paper, we focus only on routing. There are two types of packet switched networks: datagram networks and virtual circuit networks. In a connectionless datagram network, packets are forwarded along the shortest path to the destination. This approach is good only for best-effort sessions. For sessions with QoS guarantees, packets are forwarded along the path that has adequate bandwidth to support the QoS requirement. QoS routing is a special case of constraint based routing (CBR) in which computing optimal routes subject to constraints of two or more additive and/or multiplicative constraints is *NP-complete* [20]. Since calculating routes is computationally expensive, it is cost-effective to make routing decisions at the timescale of flows for long running sensornet applications.

Computing the path can be considered as the static aspect of QoS routing. The dynamic aspect (update strategy) deals with collecting network wide link state information to aid the routing algorithm. The current approach is to flood the network with updated link information. Flooding has three drawbacks; one, it is expensive; two, determining *when* to flood since there is clearly a tradeoff between the frequency of flooding and the accuracy of link state information; three, ensuring the convergence time of flooding is less than the rate at which link state changes.

Given the above scenario, we are now faced with a dilemma. On one hand we have shortest path routing that comes with minimal overhead but provides no QoS guarantees. On the other hand we have QoS routing that provides the requisite guarantees but comes with a high overhead of update strategy which in turn could affect the utilization of the network. What is required is a strategy that provides QoS guarantees, uses little overhead and provides high network utilization. In this paper we propose a strategy to do just that.

The paper is organized as follows. In Section II we present related work. In Sections III and IV, we present our system model and technical approach. Section V provides results and we conclude in Section VI.

II. Related Work

Designing a network to support QoS is thus very important and has been long recognized in the networking community. In the recent past several contributions have been made to support QoS in packet-switched networks. Notable among them are the *Integrated Services/RSVP* model [3][4], the *Differentiated Services* model [5][6], *MPLS* [7], *Traffic Engineering* (TE) [8] and *Constraint Based Routing* (CBR)[9]. QoS routing is a kind of CBR which depends upon the state of all links in the entire network and is maintained at the router's global link state database (GLSDB). A router also maintains the state of its own links in the local link state database (LLSDB). Occasionally, a router will flood the network with its LLSDB when there is a change in its link state. Each router, on receiving the LLDSB, will update its GLSDB. Reducing these link state update (LSU) messages affect the quality of routing decisions. Excessive LSU messages in turn affect the performance of routers [11]. This problem with QoS routing

can be characterized by a trade-off between the accuracy of constraint information and the overhead of exchanging LSU messages [12].

Several update algorithms have been proposed in literature. In the period based (PB) LSU algorithm [13], routers periodically transmit topology and link state information. In the threshold based (TB) LSU algorithm [13], routers generate LSU messages when the link state variation is larger than a specified threshold value (*th*). For router *i*, let bw_i^o and bw_i^c be the link state information previously advertised and currently measured respectively. Router *i* transmits LSU messages when $|bw_i^o - bw_i^c|/bw_i^c > th$. In the equal class based (ECB) LSU algorithm [14] and the unequal class based (UCB) LSU algorithm [14], a link's capacity is partitioned into classes. When the available bandwidth of a link is changed from one class to another, the corresponding router transmits LSU messages. The dynamic threshold based (DTB) algorithm [15] uses a threshold value that varies with the number of update messages generated during a fixed duration (*T*). The threshold value used in the k^{th} duration, th_k is determined as:

$$th_k = \begin{cases} th_{k-1} + \triangle th, & \text{if } \tilde{R_k} \leq R_0, \\ th_{k-1} - \triangle th, & \text{if } \tilde{R_k} > R_0 \end{cases}$$

where th_k is the threshold value used in the k^{th} duration, R_0 is the target update rate, and $\tilde{R_k}$ is the update rate during the last *T*. $\tilde{R_k}$ can be obtained as

$$\tilde{R_k} = (n_{k \cdot T}) - (n_{(k-1) \cdot T})/T$$

where $(n_{k.T})$ denotes the total number of LSU messages received up to the k^{th} *T*.

To adaptively reflect link state information, the second moment based (SB) LSU algorithm [13] introduces a stability function of two parameters; the expectation μ and the variance σ^2 of the traffic rate on a link. If the value of the stability function is equal to or greater than a specified value of *th*, LSU messages are transmitted. In the simple adaptive (SA) LSU algorithm [16], if the variation of the used bandwidth per connection is equal to or greater than the mean available bandwidth per connection expected at the instant of the next link state update, LSU messages are transmitted. The enhanced simple adaptive (ESA) LSU algorithm [17] introduces a new parameter as an impact factor to represent the effect of the connection under consideration on the entire network. Each router locally maintains the path lengths of all connections on a link, represented by *H(t)*, and the maximum value of the path length among them, represented by $H_{MAX}(t)$. The impact factor $\tilde{I(t)}$ is defined as:

$$\tilde{I(t)} = \frac{H(t)}{H_{MAX}(t)}$$

Let B_n be the available bandwidth on a link stored in the GLSDB, $\tilde{B(t)}$ be the current available bandwidth on the link and $\tilde{N(t)}$ be the number of serving connections through the link immediately prior to making the decision to transmit LSU messages. Routers transmit LSU messages if the following condition is satisfied:

$$|B_n - \tilde{B(t)}| \geq \frac{\tilde{B(t)}}{\tilde{N(t)}} \cdot \frac{1}{\tilde{I(t)}}$$

III. System Model

Instead of using a particular network as a model, we consider a network with arbitrary topologies and heterogeneous link capacity. Each link in the network has a fixed propagation delay, which is determined by the physical distance between the nodes. Transmission delay is dependent on the current capacity of the link. Queueing delay depends on the network load and the burstiness of the traffic source and the service disciplines employed in the network. A flow refers to a sequence of packets from a source to a destination and belonging to an application (session). A network flow maps to the corresponding data flow in the application's model. Thus flows between identical nodes and containing exactly the same data but belonging to different applications are considered different from each other. Before a sender sends data, it requests the network to reserve a path which guarantees the required bandwidth. If the network is able to satisfy the request, a connection is set up between the source and destination along the path selected by QoS routing. Every router along the path provides this QoS guarantee. A confirmation message is sent along the reverse path from destination to source. Upon completion of data transfer, the source sends an explicit connection tear down message. This is conceptually similar to setting up a hard-state virtual circuit as in an ATM network. Any protocol, for example RSVP, can be used to set up the connection. In this paper, we discuss only the route selection mechanism.

IV. Technical Approach

QoS routing depends upon the expensive collection of network wide LSU messages. All existing work follows the principle that at any given time all nodes in the network must have the same view of the entire network. This is why updates are flooded across the network. Accordingly, different approaches try to reduce the cost by reducing the **frequency** of updates without adversely affecting the effectiveness of the routing. However, cost can also be reduced by limiting the **size** of the flooding area. Existing approaches optimize the frequency. In this work, we will try to reduce the size of the flooding zone.

Identifying which subset of the nodes should receive the LSU messages is very difficult. If a network knew about future traffic, then it could restrict the flood region to the vicinity of the path of future sessions. Obviously, this is impossible in practical scenarios. The naive approach would be to randomly choose a subset of arbitrary size. However, this will make the impact of updates on routing decisions - well, random! Recall that establishing a session involves exchange of control information related to connection set up and tear down. This provides an opportunity to piggyback LSU messages with control messages so that nodes along the path of this session can learn about the link states of each other. Over a period of

time, with an even distribution of traffic, a node will eventually *learn* the state of the entire network which can be helpful in making decisions for future traffic. The best part is that this knowledge is free of cost! The downside of this is that the knowledge can become very stale if there is infrequent traffic to certain parts of the network. Thus all links of a downstream node might be saturated which an upstream node might not be aware of. To counter this, we propose *searching* for an alternate route starting at the node immediately preceding the saturated node and ending at the destination.

TABLE I
CONTROL PACKET

Fields			
type	Request	Confirm	Tear
flow_id	√	√	√
accepted		√	
src_path	√		
link_state <nbr_id, residual_bw>	√	√	√

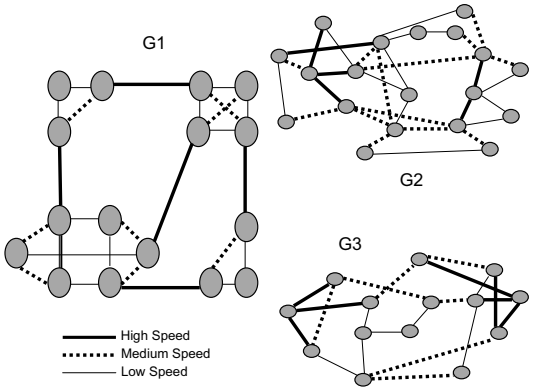

Fig. 1. Topologies used in simulation

Our approach is a classic tradeoff between computation and communication. Both computation of QoS routes and communication of LSU messages are expensive. At one end of the spectrum a solution could be to constantly flood LSU messages and compute the route once at the source. Practically this is impossible if the rate of change of traffic exceeds the rate of convergence of flooding traffic. At the other end, we could completely eliminate update messages and establish a route by doing a depth-first search of the entire network and in the process execute the routing algorithm at each hop. Instead we take a middle approach. We don't completely eliminate the update messages but make them free of cost. We use source routing whenever we can. When a path cannot be established because the stale information at the source brings us to a saturated node, we adapt to the situation by recomputing the route from that point onwards. Based on this intuition, our strategy is as follows.

Every router has its own GLSDB. At network initiation, the GLSDB has a pristine picture of the entire network. We assume all flows have unique identifiers which we refer to as the *fid*. When a edge (or first hop) router receives a connection request for a flow, it executes a shortest distance algorithm [21] using state information from its GLSDB. The shortest distance of a k-hop path P is defined as:

$$dist(P) = \sum_{i=1}^{k} \frac{1}{r_i^n}$$

where $r_1, \cdots, r_k$ are the max-min fair rates of links on the path P with k hops.

The entire route is computed at the source, and is included in the connection request packet. At each hop, a router executes an admission control policy where it checks if there is enough resources, in our case bandwidth, to support the given flow. If the flow is admitted, the requested bandwidth is set aside and the router's GLSDB and LLSDB are updated. If not, the router clears the rest of the path in the packet's source route. It then executes a shortest distance algorithm using its GLSDB to find a new path from itself to the destination. If found, then this new path is appended to the source route and the connection request is forwarded along this new path. If not found, then the router sends the request back to the immediately preceding hop on the source-computed route of the flow. Thus route computation is limited to the source and only the intermediate nodes along the source-computed path that cannot admit the flow. If the connection request reaches the destination, then a positive confirmation is sent back to the source. If instead, the request reaches the source, the flow is considered to be *blocked* and not serviceable by the network. In addition to the source, destination, *fid* and the source route, a request packet also contains a list of nodes that have already been visited to prevent routing loops. Each router maintains a flow-switching table (FST) which records the downstream router corresponding to a given flow. An entry is made in this table when a router receives a confirmation from its downstream peer. Once a source receives the confirmation, it starts sending data packets. The data packets have the *fid* instead of the destination address. Forwarding at each hop is done simply by looking up the FST. In the end, a tear down message is sent by the sender. All intermediate routers respond to this by updating their GLSDB and LLSD.

Each request, confirmation and tear down message also performs the duty of an LSU message. Specifically, when a router receives any of the above messages, it includes its LLSDB information in the packet. Recall that the LLSDB of a router has the data corresponding to its links only. At the same time, the router also updates its GLSDB with the link state information contained in the packets. If a router sees the same message more than once, for example during backtracking in connection setup, it simply updates its LLSDB information which should already be in the packet. Note that LSU messages are not piggybacked on data packets.

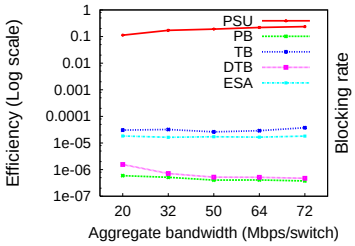

Fig. 2. Efficiency(G1)

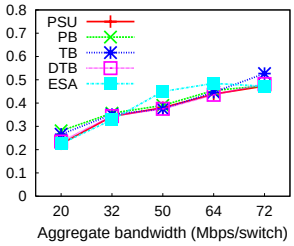

Fig. 3. Blocking rate(G1)

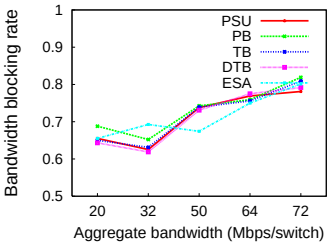

Fig. 4. Bandwidth blocking rate(G1)

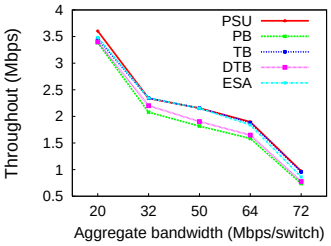

Fig. 5. Throughput(G1)

V. Evaluation

We use a two level event-driven simulation. At the session level, the simulator generates new sessions and selects routes. At the packet level, it manages control signals like connection setup and teardown, applies admission control, makes reservation and distributes link state information. Control packets are processed with highest priority so the associated delay is mainly processing delay and propagation delay. To reflect signaling delay we use connection set up and tear down costs of 3 ms/hop and 1 ms/hop, respectively, while the routing cost is 10 ms [21]. For all LSU algorithms that use network flooding, the simulator uses a simple strategy. Upon receiving an LSU message, a node updates its database and forwards the information to all its neighbors, except the one from which the update was received. Duplicate messages are discarded. We use three topologies as shown in Figure 1. Links can have a capacity of 1 Gbps (*high speed*), 500 Mbps (*medium speed*) or 100 Mbps (*low speed*). In each of the topologies, 5 to 8 host nodes are attached to each network node. The host nodes are connected with links that have capacities of either 30 Mbps or 10 Mbps.

Sessions arrive according to a Poisson distribution. We consider sessions requiring either audio or video data. The traffic source specifies a leaky bucket $\langle \sigma, b \rangle$, where σ is the token rate and b is the bucket size. The token rate determines the minimum amount of bandwidth that has to be reserved and the bucket size restricts the burst size of the traffic that is allowed to enter the network. The token rate is randomly chosen from one of the following four values: 20 Mbps that represents a single channel of High Definition resolution MPEG2 [18] encoded video, 3 Mbps that represents a PAL or NTSC-equivalent Standard Definition video, 6 Mbps that represents a multichannel DolbyDigital 'AC-3' audio with a maximum 13.1 channels [19] and 128 Kbps that represents a audio channel encoded with Advanced Audio Coding format. The maximal bucket size is uniformly distributed over one of the two intervals: [4KB,8KB] and [16KB,20KB] corresponding to streams compressed by different encoders. Exponential holding time distribution is used in most studies while others assume that sessions last forever [23][24]. We use the lognormal distribution model suggested in [25].

Metrics: The **blocking rate** [24] is the fraction of the sessions that are rejected since no path with sufficient resources can be found. The **bandwidth blocking rate** [21] is the fraction of the bandwidth that is refused a connection. The **efficiency** of the update strategy is the blocking rate per unit LSU message. The number of update messages is counted across the entire network and until the simulation ends when the last connection is completely torn down. The **average throughput** takes the weighted harmonic mean of the average per-session throughput[22], using the message length as the weight:

$$average\ throughput = \frac{\sum_{i \in N} b_i}{\sum_{i \in N} t_i}$$

where N is the total number of sessions in the network, b_i represents the number of bytes sent over connection i and t_i its duration.

We compare our approach using partial link state updates (PSU) to strategies using PB, TB, DTB and ESA approaches. All use shortest distance source routing. The X-axis in all graphs represent the average traffic that is entering the network at each network node (or router). From the graphs, it can be seen that our approach performs comparably well (within 8%) in terms of blocking rate, bandwidth blocking rate and network throughput across all the topologies. them is similar: blocking factor increases while throughput decreases with increase in traffic. The distinct advantage is in the remarkable increase in efficiency of our approach. We get similar degree of performance at a fraction of the cost of the other approaches. Consistently across all topologies, our approach is better by at least three orders of magnitude (1000%).

VI. Conclusion

In this paper, we presented a routing strategy that provides QoS guarantees, uses little overhead and provides high network utilization. Instead of using network wide flooding, we update only a part of the network with link state information. Simulation studies done over a different topologies and with realistic traffic models show that our approach is as good as any existing approach with an order of magnitude decrease in update cost.

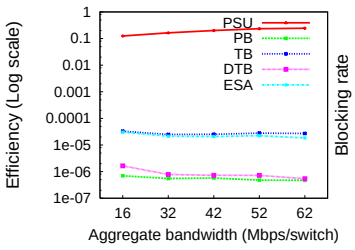

Fig. 6. Efficiency(G2)

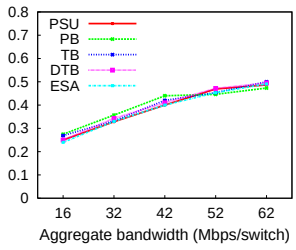

Fig. 7. Blocking rate(G2)

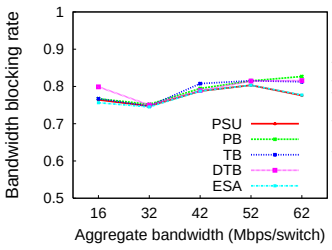

Fig. 8. Bandwidth blocking rate(G2)

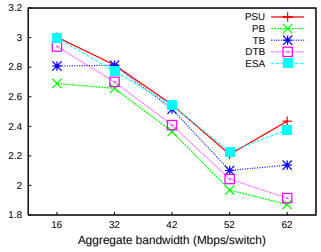

Fig. 9. Throughput(G2)

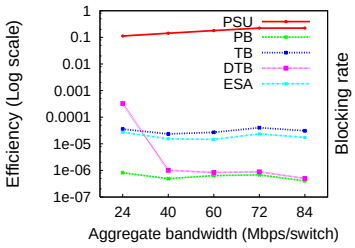

Fig. 10. Efficiency(G3)

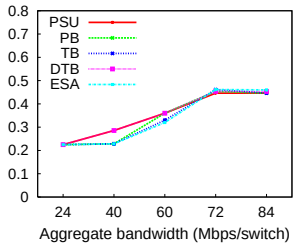

Fig. 11. Blocking rate(G3)

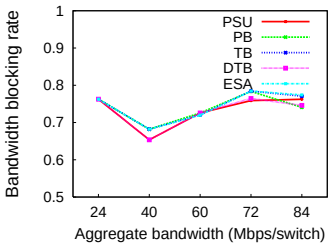

Fig. 12. Bandwidth blocking rate(G3)

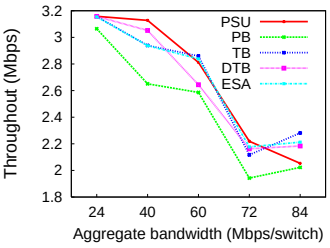

Fig. 13. Throughput(G3)

REFERENCES

[1] http://www.airport-int.com/categories/surveillance-solutions/peoplemover-project-brings-21st-century-surveillance-system-to-dallas-airport.asp
[2] http://www.tropos.com/pdf/case_studies/tropos_casestudy_new_orleans.pdf
[3] R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", Internet RFC 1633, Jun. 1994
[4] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, Sept. 1997
[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Dec. 1998.
[6] Y.Bernet et al., "A Framework for Differentiated Services", Internet draft ¡draft-ietf-diffservframework- 00.txt¿, May 1998
[7] E. Rosen, A. Viswanathan and R.Callon, "Multiprotocol Label Switching Architecture", Internet draft ¡draft-ietf-mpls-arch-01.txt¿, Mar. 1998
[8] D. Awduche, J. Malcolm, J. Agogbua, M. ODell and J. McMaus, "Requirements for Traffic Engineering over MPLS", Internet draft ¡draft-ietf-mpls-traffic-eng.00.txt¿, Oct. 1998
[9] E. Crawley, R. Nair, B. Jajagopalan and H. Sandick, "A Framework for QoS-based Routing in the Internet", RFC 2386, Aug. 1998
[10] X. Xiao and Lionel M. Ni, "Internet QoS: A Big Picture", IEEE Network, March/April 1999.
[11] G. Apostolopoulos, R. Guerin and S. Kamat, "Implementation and Performance Measurements of QoS Routing Extensions to OSPF", IN-FOCOM 1999.
[12] Y. Jia, I. Nikolaidis, and P. Gburzynski, "Multiple Path Routing in Networks with Inaccurate Link State Information, " in Proc. ICC 2001.
[13] M. Zhao, H. Zhu, Li, V.O.K. and Z. Ma, "A stability-based link state updating mechanism for QoS routing," Communications, 2005. ICC 2005. 2005 IEEE International Conference on , vol.1, no., pp. 33-37 Vol. 1, 16-20 May 2005
[14] Z. Ma, P. Zhang and R.Kantola, "Influence of link state updating on the performance and cost of QoS routing in an intranet," High Performance Switching and Routing, 2001 IEEE Workshop on , vol., no., pp.375-379, 2001
[15] A. Ariza, E. Casilari, F. Sandoval, "QoS routing with adaptive updating of link states," Electronics Letters , vol.37, no.9, pp.604-606, 26 Apr 2001
[16] S.H. Choi, M.H. Jung, M.Y. Chung, M. Yang, T. Kim, J. Park, "Simple-Adaptive Link State Update Algorithm for QoS Routing", International Conference on Computational Science (1) 2006: 969-972
[17] S.H. Choi, M.Y. Chung, M. Yang, T. Kim, J. Park, "An Enhanced Simple-Adaptive Link State Update Algorithm for QoS Routing", IEICE Transactions 90-B(11): 3117-3123 (2007)
[18] "Moving Picture Experts Group", October 2006 http://www.chiariglione.org/mpeg
[19] Dolby Laborotories Inc. www.dolby.com
[20] Z. Wang and J. Crowfort, "Quality of Service Routing for Supporting Multimedia Applications", IEEE JSAC 1996.
[21] Q. Ma, "QoS routing in the Integrated Services Networks", PhD thesis, CMU-CS-98-138, January 1998.
[22] R. Jain. "The Art of Computer Performance Analysis". Wiley 1991.
[23] S.Rampal, "Routing and end-to-end Quality of Service in Multimedia Networks", PhD thesis, North Carolina State University, August, 1995.
[24] R. Gawlick, C Kalmanek and K.G. Ramakrishnan, "Online Routing for Permanent Virtual Circuits", IEEE Infocom, 1995.
[25] V.A. Bolotin. "Modeling Call Holding Time Distribution for CCS Network Design and Performance Analysis", IEEE JSAC, 12)3):433-438, April 1994.