Per-flow Performance Analysis in Network Coding: An Approach Based on Real-Time Calculus

Huanzhong Li, Dongsong Ban, Wei Yang, Wenhua Dou
School of Computer Science
National University of Defense Technology (NUDT)
Changsha, Hunan, P.R. China, 410073
{huanzhongli, dsban, weiyang, wenhuadou}@nudt.edu.cn

***Abstract*—Network coding provides a powerful mechanism for improving performance of communication networks. In this paper, we present an analytical approach for per-flow performance analysis in network coding. Prior work on performance analysis in network coding mainly focuses on the throughput of the overall network. Our approach aims to analyze the end-to-end performance of each flow in the network. The theoretical basis of our approach is Real-Time Calculus (RTC). Under the framework of RTC, we obtain theoretical formulations for computing the backlog and delay bounds of flows in network coding. Based on the formulations, we evaluate the per-flow performance in representative scenarios and investigate the effect of coding opportunities. Numerical results show that the coding opportunities at coding nodes can significantly affect the end-to-end performance of flows in network coding.**

***Keywords*—performance analysis; network coding; real-time calculus; coding opportunities**

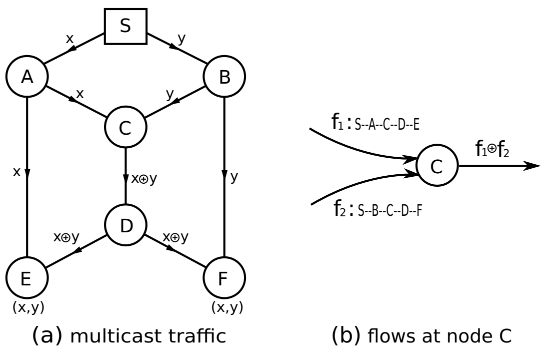


Fig. 1. The problem of per-flow performance analysis at coding nodes. Source node S sends two packets x and y to both destination nodes E and F using network coding to maximize throughput (as shown in (a)). Two flows are combined together at coding node C as depicted in (b).

I. Introduction

Network coding (see, e.g., [1], [2], [3], [4]) provides a powerful mechanism for improving throughput and robustness in communication networks [5], [6]. With network coding, a network node is not purely a packet forwarding device, but instead, it combines packets together to increase the information content delivered in a single transmission. Nevertheless, the majority of network coding research is targeted at obtaining and achieving the maximum capacity of a network [4], and few literatures discuss the end-to-end performance for traffic flows in the network. The end-to-end performance of a flow in the network is a key parameter of QoS (Quality of Service) of the network. In this paper, we try to build a analytical method for the per-flow performance analysis in network coding.

We use the classical butterfly network to explain the problem on the per-flow performance analysis in network coding. Fig. 1(a) shows the network topology and the coding strategy. Source node S wants to multi-casts two packets x and y to both destinations E and F. The packet labeled beside a link indicates what packet the link transmits, where $x \oplus y$ means the XORed packet of x and y. Note that the bottleneck node C codes the two packets x and y together and sends out only one coded packet $x \oplus y$. When the coded packet reaches its destinations, the destinations E and F can decode their expected packet accordingly. We emphasize that the coded packet $x \oplus y$ outgoing from node C carries the information content of two packets x and y, leading to throughput increases for the multi-cast communication.

Consider the coding node C. There are two conventional questions on the per-flow performance analysis at C: how many buffer used by each flow at the node; how long the node incurs packet delays for each flow. To answer these questions, we have an elegant theory named *Network Calculus*. Network Calculus is a theory dealing with queuing systems found in computer networks[7]. It is first proposed by Cruz [8], and largely used in the performance analysis of communication networks (see [9], [10] and references therein). We can use Network Calculus to calculate the upper bounds of a flow's delay and the maximum buffer required by the flow at a node. However, in networks with coding, Network Calculus can not take the throughput increments caused by coding nodes into account, which makes the theory inapplicable.

In Network Calculus, a basic assumption is that multiple flows at a node share the service of the node separately. Based on this assumption, we have a leftover service theorem for each flow, which says that the service provided for a specific flow equals the total service of the node minus the service assigned to all other cross flows. However, if we consider a coding node, the leftover service theorem can only derive an underestimated service for the flow. Take node C in Fig. 1 for an example, if we consider the service provided for flow f_1, then flow f_2 becomes the cross flow of f_1. However, we need not to subtract all the service assigned to the cross flow f_2, because it is possible that packets of f_1 can be combined and sent out together with packets of f_2. It is viewed like that f_1 can also takes a free ride on f_2. In other words, f_1 not only gets the service assigned by the node, but also captures extra service from cross flows.

A method to analyze the per-flow performance at a coding node is to model the extra service captured from cross flows. To this end, we introduce the theory of *Real-Time Calculus* (RTC) [11], [12], [13]. RTC extends the concepts of Network Calculus to characterize more properties of flows and service. We use the *arrival curve* of RTC to model the cross flows, which can be used to calculate the extra service that the cross flows offer. In this model, the overall service provided for a flow at a coding node equals the sum of the assigned service from the node and the extra service from cross flows. We further discuss this model in Section II.

A practical problem is how to calculate the extra service from cross flows in the model based on RTC. Note that the extra service is determined by the amount of cross traffic that can provide a free ride for the considered flow. If we properly define the *coding opportunities* at coding nodes, we can compute the extra service directly. For example, if every packet of cross flows can provide a free ride for the considered flow, then the coding opportunities equals 100%. In this case, it seems that the considered flow catches all the service of the node. We further discuss this topic in Section III.

Our research makes several contributions as follows. We make a forward step on per-flow performance analysis in network coding, which clarifies that the problems lie in the assumption of separate service share among flows. Furthermore, by introducing RTC, we build an analytical method to analyze the per-flow performance. By investigate the numerical results of out method, we find that the coding opportunities at coding nodes can significantly affect the end-to-end performance of flows in network coding. These theoretical results can help to design and improve a practical network coding protocol.

The rest of this paper is organized as follows. In Section II, we introduce the basis of Real-Time Calculus (RTC) and construct a new theorem for per-flow performance analysis in network coding. In Section III, we define the coding opportunities at a coding node and then derive some theoretical results based on the definition. In Section IV, we evaluate performance of a representative scenario and investigate the effect of coding opportunities in network coding. We conclude the paper at Section V.

II. Real-Time Calculus Model

In this section, we first introduce the theory of Real-Time Calculus (RTC) that we use to analyze the per-flow performance in network coding. And then, we propose a new theorem based on RTC to support the service analysis at a coding node.

A. Real-Time Calculus Basis

In RTC, a flow is described by means of a cumulative function $A(t)$, where $A(t)$ denotes the amount of traffic that have already arrived at time t. Similarly, the service provided for a flow is also described by means of a cumulative function $S(t)$, where $S(t)$ denotes the amount of service that the node can provide for the flow up to time t. Conventionally, we consider the flow starting from the time $t = 0$, i.e., $A(t) = S(t) = 0$. The *arrival curve* and the *service curve* are two fundamental concepts in RTC. They characterize the properties of the arriving flow and the service provided for the flow.

Definition 2.1 (Arrival Curve): Consider a flow $A(t)$. If there exists two functions $\alpha^l(t)$ and $\alpha^u(t)$ that satisfy

$$\alpha^l(t-s) \leq A(t)-A(s) \leq \alpha^u(t-s) \quad \text{for all } 0 \leq s \leq t, \tag{1}$$

where $\alpha^l(t) = \alpha^u(t) = 0$, we say the flow has an arrival curve $< \alpha^l, \alpha^u >$, where α^l is the lower arrival curve, and α^u is the upper arrival curve.

Definition 2.2 (Service Curve): Consider a flow passing through a node. Let $S(t)$ denote the service that the node provides for the flow. If there exists two functions $\beta^l(t)$ and $\beta^u(t)$ that satisfy

$$\beta^l(t-s) \leq S(t)-S(s) \leq \beta^u(t-s) \quad \text{for all } 0 \leq s \leq t, \tag{2}$$

where $\beta^l(t) = \beta^u(t) = 0$, we say the node provides a service curve $< \beta^l, \beta^u >$ for the flow, where β^l is the lower service curve, and β^u is the upper service curve.

With the definition of arrival curve and service curve, we have the following theorems for performance analysis.

Theorem 2.3 (Performance Bounds): Consider a flow passing through a node. If the flow has an arrival curve $< \alpha^l, \alpha^u >$ and the node provides a service curve $< \beta^l, \beta^u >$ for the flow, then the delay $d(t)$ of the flow at time t is bounded by

$$d(t) \leq h(\alpha^u, \beta^l), \tag{3}$$

where $h(\alpha^u, \beta^l)$ is the maximum horizontal distance between $\alpha^u(t)$ and $\beta^l(t)$, defined as

$$h(\alpha^u, \beta^l) = \underset{s \geq 0}{sup}\Big\{inf\{\tau \geq 0 : \alpha^u(s) \leq \beta^l(s+\tau)\}\Big\};$$

the backlog $b(t)$ of the flow at time t is bounded by

$$b(t) \leq v(\alpha^u, \beta^l), \tag{4}$$

where $v(\alpha^u, \beta^l)$ is the maximum vertical distance between $\alpha^u(t)$ and $\beta^l(t)$, defined as

$$v(\alpha^u, \beta^l) = \underset{s \geq 0}{sup}\Big\{\alpha^u(s) - \beta^l(s)\Big\}.$$

B. A Key Theorem for Performance Analysis

In order to use RTC to analyze the performance of a flow at a coding node, we need to know the arrival curve of the flow and the service curve that the node provides for the flow. For the arrival curve, there is no difference between arrival curves with and without coding, because they depend only on the traffic of the flow. So, we can derive the arrival curve as convention. However, for the service curve, since the service provided for the flow consists of the assigned service from the node and the extra service from cross flows, we have to add up these two service before applying Theorem 2.3 to calculate the performance bounds.

We propose a new theorem for calculating the overall service of a coding node, which is the key theorem for the per-flow performance analysis in network coding.

Theorem 2.4 (Adding Up Service): Consider a flow is simultaneously served by two servers in parallel. Suppose that one server provides a service curve $< \beta_1^l, \beta_1^u >$ for the flow, and the other server provides a service curve $< \beta_2^l, \beta_2^u >$ to the flow. Then the overall service curve provided by these two servers is $< \beta_1^l + \beta_2^l, \beta_1^u + \beta_2^u >$.

Proof: With the definition of service curve we have

$$\beta_1^l(t-s) \le S_1(t) - S_1(s) \le \beta_1^u(t-s)$$
$$\beta_2^l(t-s) \le S_2(t) - S_2(s) \le \beta_2^u(t-s),$$

where $S_1(t)$ and $S_2(t)$ are the service provided by the two servers respectively. By adding up these two inequalities, we must have

$$(\beta_1^l+\beta_2^l)(t-s) \le (S_1+S_2)(t)-(S_1+S_2)(s) \le (\beta_1^u+\beta_2^u)(t-s),$$

which concludes the proof.

Remark: Theorem 2.4 looks so simple that someone may doubt its importance. We emphasize that the concise results of Theorem 2.4 is rooted in the definition of service curve in RTC. If we use the definition of service curve in Network Calculus, no result on adding up service can be derived. Besides, we have another benefit of using RTC when calculating the service curve of extra service (see Section III).

III. Performance Analysis with Coding Opportunities

In network coding, the arrival curve of a flow can be derived as usual (see Subsection II-B). However, when deriving the service curve of a coding node, we encounter a new problem. Generally, we can derive the service curve of the assigned service from a node, but we have no theorem to derive the service curve of the extra service from cross flows. In this section, we propose a method to solve this problem, and then present the main results of per-flow performance analysis in network coding.

A. Computing Extra Service

The extra service from cross flows depends on how much traffic of the cross flows that can provide free rides for the considered flow. We wish to measure the amount of such ridable traffic so that we can compute the service curve of the extra service. We define *coding opportunities* that indicates the proportion of ridable traffic in a flow.

Definition 3.1 (Coding Opportunities): Consider two flows f_1 and f_2. Let $A_2(t)$ denote the traffic amount of f_2. If the amount of ridable traffic of f_2 for f_1 equals $p_{21}A(t)$, then we say that f_2 has coding opportunities of p_{21} for f_1.

With the definition of coding opportunities, we then have the following theorem to derive the service curve of extra service.

Theorem 3.2 (Extra Service): Consider two flows f_1 and f_2. Suppose that f_2 has an arrival curve $< \alpha_2^l, \alpha_2^u >$, and that f_2 has coding opportunities of p_{21} for f_1. If the two flows pass a coding node with enough service capacity, then the extra service that f_2 provided for f_1 has a service curve of $< p_{21}\alpha_2^l, p_{21}\alpha_2^u >$.

Proof: With the definition of arrival curves we have

$$\alpha_2^l(t-s) \le A_2(t) - A_2(s) \le \alpha_2^u(t-s),$$

where $A_2(t)$ is the traffic amount of f_2. By multiplying this inequality with p_{21}, we then have

$$p_{21}\alpha_2^l(t-s) \le p_{21}A_2(t) - p_{21}A_2(s) \le p_{21}\alpha_2^u(t-s).$$

Note that, in a node with enough service capacity, the ridable traffic of f_2 can bring out the same amount of its riding flow f_1. And so, the the extra service $S_{21}(t)$ that f_2 provided for f_1 equals the amount of ridable traffic of f_2, i.e., $S_{21}(t) = p_{21}A_2(t)$. It follows that $p_{21}\alpha_2^l(t-s) \le S_{21}(t) - S_{21}(s) \le p_{21}\alpha_2^u(t-s)$, which concludes the proof.

Remark: The concise results of Theorem 2.4 is also rooted in the using of RTC, where the definitions of arrival curve and service curve are theoretically same. It is such kind of definitions that support to derive service curve from arrival curve. If we use the definition of service curve in Network Calculus, also no result on extra service curve can be derived.

B. Per-flow Performance Analysis

So far, we can derive the arrive curve of a flow and the service curve provided for the flow. By applying Theorem 2.3, we next present the main result of our method that can analyze per-flow performance in network coding.

Theorem 3.3 (Performance Bounds with Network Coding): Consider two flows f_1 and f_2 passing through a coding node. Suppose that the two flows have arrival curves $< \alpha_1^l, \alpha_1^u >$ and $< \alpha_2^l, \alpha_2^u >$ respectively, and that the node provides a service curve $< \beta^l, \beta^u >$ for the aggregate flow. If f_2 has coding opportunities of p_{21} for f_1, then the delay $d_1(t)$ of f_1 at time t is bounded by

$$d_1(t) \le h\big(\alpha_1^u, [\beta^l - \alpha_2^u]^+ + p_{21}\alpha_2^l\big), \tag{5}$$

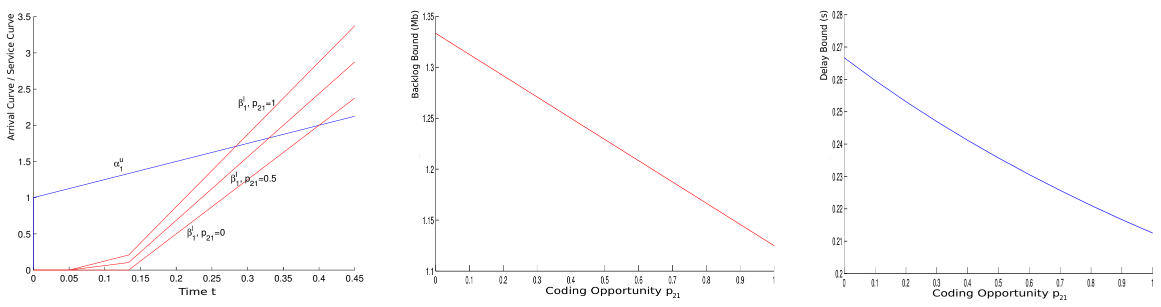
where $[x]^+ = max(0, x)$; and the backlog $b_1(t)$ of f_1 at time t is bounded by

$$b_1(t) \le v\big(\alpha_1^u, [\beta^l - \alpha_2^u]^+ + p_{21}\alpha_2^l\big). \tag{6}$$

Proof: With Theorem 2.3, we only need to prove the lower service curve β_1^l provided for f_1 is $[\beta^l - \alpha_2^u]^+ + p_{21}\alpha_2^l$.

With the leftover service theorem in Network Calculus (see Theorem 2.27 in [7]), we know that the lower service curve β_{1ass}^l of assigned service from the node for f_1 is $[\beta^l - \alpha_2^u]^+$, i.e., $\beta_{1ass}^l = [\beta^l - \alpha_2^u]^+$. We then wish to prove that the lower service curve β_{1ext}^l of extra service from the cross traffic f_2 for f_1 is $p_{21}\alpha_2^l$. According to Theorem 3.2, when the node has not been congested by flows, all the ridable traffic amount of f_2 can provide extra service for f_1, and so $\beta_{21}^l = p_{21}\alpha_2^l$.

According to Theorem 2.4, we have that the overall service curve provided for f_1 equals the sum of the assigned service curve and the extra service curve, i.e., $\beta_1^l = \beta_{1ass}^l + \beta_{1ext}^l = [\beta^l - \alpha_2^u]^+ + p_{21}\alpha_2^l$, which concludes the proof.



(a) Arrival Curve and Service Curves (b) Backlog Bound (c) Delay Bound

Fig. 2. Numerical results of per-flow performance bounds in network coding. As coding opportunities increase, the overall service curve provided for a flow at a coding node is lifted up as shown in (a), and backlog and delay bounds of the flow are both lowered down as shown in (b) and (c).

Remark: Theorem 3.3 is an extended version of Theorem 2.3. If the coding opportunities $p_{21} = 0$, then Theorem 3.3 becomes Theorem 2.3. The coding opportunities p_{21} in the theorem reflects the throughput increase caused by network coding at the coding node. We can see that the coding opportunities can significantly affect the end-to-end performance of flows (see Section IV).

IV. Numerical Results

In this Section, we use the representative arrival curve and service curve to illustrate the numerical results of our method. By investigating the results of backlog and delay in the representative scenario, we find that the coding opportunities at coding nodes can significantly affect the end-to-end performance of flows in network coding.

Consider two flows f_1 and f_2 passing through a coding node with constant service rate (CSR) as shown in Fig. 1 (b). Suppose the two flows have $< \alpha_1^l, \alpha_1^u >$ and $< \alpha_2^l, \alpha_2^u >$ as arrival curves respectively, where $\alpha_i^l(t) = \rho_i(t - T_i)$ and $\alpha_i^u(t) = \rho_i t + \sigma_i$ $(i = 1, 2)$. Since the node is a CSR node, we know that the aggregate service curve $< \beta^l, \beta^u >$ equals $< rt, rt >$, where r is the service rate of the node. By applying Theorem 3.3, we can calculate the delay and backlog bounds of f_1.

Fig. 2 shows the numerical results of our evaluations. In Fig. 2(a), we see that the overall service curve provided for f_1 is lifted up as the coding opportunities p_{21} increases. Since the delay and backlog bounds are the horizontal and vertical distance between the arrival curve and service curve respectively, these bounds becomes lower as p_{21} increases. Fig. 2(b) and Fig. 2(a) depict the backlog and delay bound with different p_{21}. In Fig. 2(b), When compared with the bound without coding ($p_{21} = 0$), the bound with perfect coding opportunities ($p_{21} = 1$) can save up to 200 Kb of buffer requirement, which is approximately 15% of the buffer assigned for the flow. Similarly, In Fig. 2(c), perfect coding opportunities can reduce up to 54 ms of delay at the node, which is approximately 20% of the packet delay experienced by the flow without coding.

With the theoretical results above, we draw a conclusion that the coding opportunities can significantly affect the end-to-end performance of flows in network coding.

V. Conclusion

In this paper, we propose a theoretical method based on Real-Time Calculus (RTC) to analyze the per-flow performance in network coding. Under the framework of RTC, we construct a new theorem that can calculate the overall service curve provided for a flow by a coding node. By properly defining coding opportunities at the coding node, we derive the theoretical bounds on backlog and delay of the flow at the node. Numerical results show that the coding opportunities at coding nodes can significantly affect the end-to-end performance of flows in network coding.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[4] R. W. Yeung, *Information Theory and Network Coding*. Springer-Verlag, 2008.

[5] T. Ho and D. S. Lun, *Network Coding: An Introduction*. Cambridge University Press, 2008.

[6] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Foundations and Trends® in Networking*, vol. 2, no. 1, p. 133, 2007.

[7] Y. Jiang and Y. Liu, *Stochastic Network Calculus*. London: Springer-Verlag, 2008.

[8] R. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE transactions on information theory*, vol. 37, no. 1, pp. 114–131, 1991.

[9] J.-Y. L. Boudec and P. Thiran, *NETWORK CALCULUS: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2001.

[10] C. Chang, *Performance guarantees in communication networks*. Springer Verlag, 2000.

[11] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. 6th Design, Automation and Test in Europe (DATE)*. Citeseer, 2003, pp. 190–195.

[12] M. G. A. M. L. Thiele, S. Chakraborty and J. Greutert., "Embedded software in network processors - models and algorithms," in *Proceedings of the 1st International Workshop on Embedded Software (EMSOFT)*, 2001, p. 416434.

[13] S. C. L. Thiele and M. Naedele., "Real-time calculus for scheduling hard real-time systems," in *International Symposium on Circuits and Systems (ISCAS)*, vol. 4, 2000, p. 101104.