On the queueing behavior of inter-flow asynchronous network coding

Y. Yuan a, K. Wu b, W. Jia c, Y. Peng a

a National Lab for Parallel and Distributed Processing, National University of Defense Technology, Changsha, China
b Dept. of Computer Science, University of Victoria, B.C., Canada
c Dept. of Computer Science, City University of Hong Kong, Hong Kong SAR, China

ARTICLE INFO

Article history:
Received 17 March 2011
Received in revised form 30 March 2012
Accepted 24 April 2012
Available online 3 May 2012

Keywords:
Performance evaluation
Network coding
Queuing analysis
Congestion control

ABSTRACT

Despite the substantial research efforts on network coding, its real-world implementation is mainly over wireless networks or peer-to-peer networks. The deployment of network coding in the Internet core still largely lags behind. Among the many challenges, one difficulty is the selection of routers to perform network coding, which relies on the understanding of the queueing behavior of network coding. Unfortunately, the intricate queueing behavior of network coding, even for a single node case, is still unclear.

In this paper, we build a generic queueing model to answer many fundamental questions, including for example, under what condition is the system stable? How many packets could be possibly coded when multiple stochastic traffic flows pass through a coding node? What is the quantitative relationship among the traffic arrival rate, the service rate, and the coding opportunities under a general network configuration? Based on our analytical results, we propose a self-adjustable delay-based coding mechanism for better congestion control. Our work provides network researchers and engineers with insights on the queueing behavior of network coding, which are helpful in future applications of network coding in the Internet core.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The concept of network information flow was introduced in 2000 [1], and since then network coding has triggered enormous research and development activities. In the existing Internet architecture, network devices such as routers and switches mainly function as store-and-forward relay devices. Network coding technique changes the above fundamental design principle and allows the algebraic combination of data packets at intermediate network nodes by using, for example, a bit-by-bit XOR operation. Destination nodes perform decoding operations to recover the original data packets.

In theory, it has been proved [1] that network coding can achieve the maximum information flows for multicast, which are otherwise impossible for traditional store-and-forward networks. In practice, network coding has been implemented and tested in different systems, particularly in wireless networks where radio transmission is broadcast in nature and a node can overhear neighbors’ transmissions [2]. Network coding has also found its success in peer-to-peer networks for content distribution [3]. Extra benefits of network coding include reducing energy consumption for wireless networks, enhancing network security and reliability, reducing bandwidth cost for content distribution over P2P networks, and much more.

With the practical success of network coding in wireless networks, it becomes a natural and important question how network coding can be deployed in the Internet core. This question cannot be simply answered because network coding requires substantial changes on current router architecture and existing protocols for traffic control. Among the many challenges, one difficulty relies on the subgraph selection in large networks, i.e., the selection of routers to perform network coding. To illustrate this, an example using the broadly cited “butterfly” network in the Internet core is depicted as follows:

Example 1. As illustrated in Fig. 1, flow 1 (S1 → D1) and flow 2 (S2 → D2) share the link from R1 to R2. Suppose every link is lossless. Due to the heavy traffic from S1 and S2, router R1 may eventually get congested (Fig. 1(a)). To reduce the traffic load, R1 first notifies S1(S2) to route its packets to D2(D1) via redundant path 1(2), then XORs the buffered packet pairs from each flow and transmits the coded packets to router R2 (Fig. 1(b)). Note that D1 or D2 can decode the coded packets by calculating the XOR with its corresponding original packets from the redundant path. However, since the two flows arriving at R1 may not be well synchronized, R1 may only buffer packets from one of the two flows, such that the coding opportunities among the two flows may not always exist.
2.1. Synchronous network coding

In synchronous network coding, a coding node holds the coding operation until all required coding packets from different flows arrive. In the early stage of network coding, most of the code design algorithms to achieve the maximum network capacity are based on the implicit assumption that the packets from different input flows can be well synchronized. In real-world networks, however, stochastic property of the input traffic makes the queueing behavior of network coding much more complex. In [19], Ma et al. studied a queueing model for synchronous network coding with two identical Poisson input flows. They obtain the result that the output flow will be an asymptotically Poisson flow with the same parameter. In [20], Ma et al. extended their previous work, and found that with synchronous network coding the system becomes unstable with the queue size increasing to infinity. To handle this, they then propose an COS strategy with synchronous coding stage and clearing stage, which can be regarded as a scheme of asynchronous network coding.

The queueing model of synchronous network coding can find its root in assembly-like manufacturing operations. Early in 70s, the assembly-like queueing model was first studied in [4]. An important result is that when the buffer size is infinitely large, the queueing system is unstable in the sense that the buffer will eventually overflow unless the input flows are deterministic and fully synchronized [4]. An analytical method for assembly-like queue with two Poisson input flows and finite queue capacity was proposed in [5]. The author concluded that exact analysis became intractable if the number of input flows was large. In [21], the authors used the difference between the accumulative traffic amount of each class and the minimum accumulative traffic amount in these classes to find an approximate solution. Furthermore, the authors use a decomposition technique to obtain the approximation solution for multiple Poisson flows. Another approximate solution was proposed in [22]. The approximation is based on the idea that a flow is “turned off” whenever it arrives too fast compared to other flows. In [23], the authors proved that the assembly-like queue with two input flows is stochastically equivalent to a transfer line of tandem queues with blocking. Although this approximation requires little computation compared to those in [5,22,21], it works only when there are two input flows.

2.2. Asynchronous network coding

In asynchronous network coding, a node may transmit uncoded packets or partial coded packets, if no coding opportunities for all the required packets could be found within a bounded time window. The asynchronous network coding makes more practical sense and was first implemented in COPE [2]. Its performance was mainly evaluated with measurements. Since then, many schemes based on asynchronous network coding was proposed to achieve better performance [10,24–26]. For example, in [10], two practical code construction techniques based on linear optimization were proposed; in [24], Eryilmaz et al. used differential backlogs to decide the coding opportunity for inter-flow network coding; in [25], An online scheme combining the benefits of network coding and ARQ for lossy channels was proposed; in [26], Seferoglu et al. designed a “network-coding aware” queue management scheme (NCAQM) at intermediate nodes to fully exploit the coding opportunities.

Although substantial efforts have been devoted to the performance analysis of asynchronous network coding [6,7,27–29], the problem was addressed with different assumptions or in different contexts, e.g., in wireless networks with random medium access [6], or in the study of energy consumption of asynchronous network coding with two Poisson input flows [7], or on the delivery of real-time packets [27] from two Poisson input flows, or for random linear coding over packet erasure channels [28]. In [29], the authors investigated the similar problems using stochastic network calculus. Their results only disclose the performance bounds in idealized situations, which may be practically hard to achieve. Although the work in [30] bears some similarities to our approach, it only considers the queueing behavior of synchronous network coding and the special situation of asynchronous network coding that the maximum opportunistic delay is set to zero.

Up to now, we do not see any formal model and mathematical analysis answering the questions raised in Section 1 in a general setting.

3. Queueing model of asynchronous network coding

In this section, we present a queueing model for asynchronous network coding. As shown in Fig. 2, packets from n flows arrive into a router which can perform network coding (coding node). Besides the coding rules introduced in Example 1, we assume the following configurations:

- Traffic arrivals: Incoming flows are assumed to be independent. We assume that the packet inter-arrival times of flow $i(=1, \ldots, n)$ are independent identically distributed (i.i.d.) random variables with mean arrival rate $\lambda_i$. Denote the distribution function of packet inter-arrival times of flow $i(=1, \ldots, n)$ by $A_i(t)$.
- Buffer: The coding node maintains a separate first-in-first-out (FIFO) queue with a queue size $K(+\infty)$ for each flow. The queue corresponding to flow $i(=1, \ldots, n)$ is labeled with index $i$.
- Service: Assume that the service time for coding and transmitting the (coded) packets follows a general service time with mean service rate $\mu_i$. Denote the distribution function of the service time as $B(t)$.
- Maximum opportunistic delay: To increase coding opportunities, the coding node may wait up to the maximum opportunistic delay, $D$. When $D \to \infty$, the asynchronous network coding is equivalent to the synchronous network coding. In this paper, we only focus on the asynchronous network coding, i.e., $0 \leq D < \infty$.
- Links: The links are assumed to be lossless and reliable.
- Network code: We assume that the network code is properly designed (random or deterministic linear network coding). Extra bits are piggybacked in each coded packet for identifying its original components, such that the coded packets can be decoded at the intended receiver(s). Network coding can be performed on a group of $k$ packets, where $1 \leq k \leq n$. The value of $k$ is determined by the number of nonempty queues when the router performs network coding.

![Fig. 2. Queueing model of asynchronous network coding.](image-url)
Our method could be extended to the situation when asynchronous network coding is required. The difficulties come from the complex correlation among the variables. When maximum allowed delay is set on packets, suitable results, we need to solve a set of n equations with n unknown variables. When maximum allowed delay is set on packets, suitable values for the maximum opportunistic delay should be obtained first. This forms another interesting question which needs to be handled carefully.

### 4. Performance analysis of asynchronous network coding

Exact solution using state transition diagram for synchronous network coding turn out to be extremely hard [29]. To say nothing of asynchronous network coding in the network wide, even for a special case where incoming packets are all considered innovative, it could be extended to deal with other coding schemes by building a bridge between information and packets [35]. We leave it as our future work.

#### Remark 2.

In some network coding implementation schemes (particularly random network coding), coding decision may depend on the innovative incoming packets. In this case, information of data packets is considered and thus special treatment is required to bridge the gap between data traffic and its information [35]. Nevertheless, the abstract model in this paper could be considered as a special case where incoming packets are all considered innovative, and it could be extended to deal with other coding schemes by building a bridge between information and packets [35]. We leave it as our future work.

#### Remark 1.

There are many other factors that may impact the coding opportunity of network coding, such as the unreliable transmissions [31,32], MAC layer contention [6], the coordination between the senders [33], and whether the incoming packets are innovative [34]. Nevertheless, taking into consideration of all these factors will make the system intractable. As shown in the paper, even with the above simplified configuration, modeling the queueing behavior of network coding is already very difficult.

### 4.1. Analysis of coding opportunities

For ease of reference, the main notations used in our analysis are listed in Table 1.

#### Definition 1 (Coding opportunity).

Viewed from the time instances when a (coded) packet is sent out from the router, the head packet of a nonempty queue, if exists, is said to have a coding opportunity if there are other nonempty queues or if there are packet arrivals to other empty queues within the maximum opportunistic delay $D$. Whenever a coding opportunity exists, the head packet of a nonempty queue should be coded together with the head packets in other nonempty queues. If a packet is coded with other $l$ packets, we call the packet is $l$-level coded.

#### Definition 2 (Coding opportunity matrix). The coding opportunity matrix of asynchronous network coding with $n$ input flows is a $n \times n$ matrix, denoted by

$$
P = \begin{bmatrix}
    p_0^0 & p_1^1 & \cdots & p_{n-1}^{n-1} \\
p_0^l & p_1^l & \cdots & p_{n-1}^l \\
\vdots & \vdots & \ddots & \vdots \\
p_0^n & p_1^n & \cdots & p_{n-1}^n
\end{bmatrix},
$$

where $p_i^j$ denotes the probability that the head packet of queue $i$ is $j$-level coded (i.e., it is coded with exactly $j$ packets from other flows). The matrix also can be written as column vectors: $\mathbf{P} = [\mathbf{C}^0, \mathbf{C}^1, \ldots, \mathbf{C}^{n-1}]$, where $\mathbf{C}(0 \leq j < n) = \{p_0^j, p_1^j, \ldots, p_n^j\}^T$.

We first calculate the coding opportunity matrix, which will be used later to decompose the queueing model and to obtain the performance of interests. In order to calculate $P$, we need to obtain at the instances when a (coded) packet is transmitted from the router, (1) the steady-state probability that a queue is empty, and (2) the probability that there are packet arrivals to the empty queue(s) within the time period of $D$.

#### When considering $P$, we call queue $i$ the tagged queue, meaning that when the previous service is over (i.e., a packet is sent out from the router), queue $i$ has packets to be served. From the point of view of the tagged queue, other $n - 1$ queues may either have packets or empty at this instance. Let $E_i$ denote the steady state probability that queue $j$ is empty at this instance. We will introduce its calculation in the next section.

Because the packet inter-arrival times follow a general i.i.d. distribution, the memoryless property of Markov arrival process may not hold. To calculate the probability that there are packet arrivals to an empty queue within the time period of $D$, we need to consider the residual arrival time of the next incoming packet to the empty queue. Let $Y_j$ denote the residual arrival time of the next incoming packet to queue $j$. The probability that there are packet arrivals to queue $j$ within the period of $D$, $F_j$, can be calculated as [36]:

$$F_j = \Pr\{Y_j \leq D\} = \int_0^D \left(1 - A_i(y)\right) \frac{1}{\lambda_i} \, dy. \quad (1)$$

#### Table 1

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_i$</td>
<td>The packet inter-arrival time of flow $i$ (a random variable)</td>
</tr>
<tr>
<td>$A_i(t)$</td>
<td>The distribution function of $a_i$</td>
</tr>
<tr>
<td>$\text{Var}(a_i)$</td>
<td>The variance of $a_i$</td>
</tr>
<tr>
<td>$\lambda_i$</td>
<td>The mean arrival rate of flow $i$</td>
</tr>
<tr>
<td>$B(t)$</td>
<td>The distribution function of the service time of the router</td>
</tr>
<tr>
<td>$\mu$</td>
<td>The mean service rate of the router</td>
</tr>
<tr>
<td>$D$</td>
<td>The maximum opportunistic delay</td>
</tr>
<tr>
<td>$Y_i$</td>
<td>The residual arrival time of the next incoming packet of flow $i$ (a random variable)</td>
</tr>
<tr>
<td>$\bar{P}_i$</td>
<td>The distribution function of $Y_i$</td>
</tr>
<tr>
<td>$P_i^j$</td>
<td>The probability that the head packet of queue $i$ is $j$-level coded within the maximum opportunistic delay $D$</td>
</tr>
<tr>
<td>$E_j$</td>
<td>The steady state probability that queue $j$ is empty when queue $i$ has packets waiting for service, after a (coded) packet is sent</td>
</tr>
<tr>
<td>$E_i^{(1)}$</td>
<td>The approximation of $E_i$ (Section 4.2)</td>
</tr>
<tr>
<td>$Q_i^{(1)}$</td>
<td>The index set of all queues excluding queue $i$</td>
</tr>
<tr>
<td>$Q_i^{(k)}$</td>
<td>A subset of $Q_i^{(k-1)}$, formed by taking any $k$ elements from $Q_i^{(k-1)}$</td>
</tr>
<tr>
<td>$S_i(t), a_i$</td>
<td>The distribution function and the pdf of the opportunistic delay in sub-system $i$, respectively</td>
</tr>
<tr>
<td>$D_i(t)$</td>
<td>The distribution function of the “service” time of the virtual server in sub-system $i$</td>
</tr>
<tr>
<td>$r_i(t)$</td>
<td>The total service time of sub-system $i$ (a random variable)</td>
</tr>
<tr>
<td>$R_i(t)$</td>
<td>The distribution function of $r_i$</td>
</tr>
<tr>
<td>$\text{Var}(r_i)$</td>
<td>The mean and the variance of $r_i$, respectively</td>
</tr>
<tr>
<td>$\iota_i$</td>
<td>The idle period of sub-system $i$ (a random variable)</td>
</tr>
<tr>
<td>$\iota_i^2$</td>
<td>The first and the second moments of $\iota_i$, respectively</td>
</tr>
</tbody>
</table>
With \( E_\varphi \) and \( \hat{F}_j \) at hand, we can calculate \( P_j \) as follows:

\[
P_j = \sum_{k=0}^{n-1} \left\{ \prod_{i=1}^{\hat{F}_j} \left( 1 - \frac{h_n}{E_i} \right) \prod_{q_{k,j}=1}^{\hat{F}_j} \left( 1 - \frac{h_{q_{k,j}}}{E_i} \right) \right\},
\]

where \( Q_k^j \) denotes the set of index of the \( n-j \) queues except queue \( i \), \( Q_i^j \) denotes a subset of \( Q_k^{n-1} \) by selecting any \( k \) elements from \( Q_k^{n-1} \), \( Q_i^{n-1} \) denotes the set of index of the \( n-1-k \) empty queues given the number of nonempty queues is \( k \), and \( Q_i^{n-1} \) denotes a subset of \( Q_k^{n-1} \) and represents those empty queues that have packet arrivals within the period of \( D \).

The lengthy equation (2) is actually a concise presentation of all possible combinations in the calculation of \( P_j \). For easy understanding, we use an example to illustrate the calculation.

**Example 2.** Assume that a router has three independent input flows. The corresponding queues are labeled by queue 1, 2, and 3, respectively. We first consider \( P_{1,l} \). In this case, \( k \) in the first \( \Sigma \) of Eq. (2) can only be zero; \( Q_1^0 = \{2, 3\} \). \( Q_1^0 \) is an empty set. From Eq. (2), we thus obtain

\[
P_{1,l} = E_{12} \cdot E_{13} \cdot (1 - \hat{F}_2) \cdot (1 - \hat{F}_3).
\]

The meaning of Eq. (3) is clear: the probability that the head packet of queue 1 has no coding opportunity is equal to the probability that the other two queues are empty and during \( D \), no packets arrive at the other two queues.

Similarly, we can obtain \( P_{1,1} \) and \( P_{1,2} \) as follows:

\[
P_{1,1} = E_{12} \cdot (1 - E_{13}) \cdot (1 - \hat{F}_2) + (1 - E_{12}) \cdot E_{13} \cdot (1 - \hat{F}_3) + E_{12} \cdot E_{13} \cdot (1 - \hat{F}_2) \cdot \hat{F}_3,
\]

\[
P_{1,2} = E_{12} \cdot E_{13} \cdot \hat{F}_2 \cdot \hat{F}_3 + (1 - E_{12}) \cdot E_{13} \cdot \hat{F}_3 + E_{12} \cdot E_{13} \cdot (1 - \hat{F}_2) \cdot \hat{F}_3 + (1 - E_{12}) \cdot (1 - E_{13}) \cdot \hat{F}_2 \cdot \hat{F}_3.
\]

The calculation of other values in \( P \) is similar.

Next, we discuss some interesting properties of the coding opportunity matrix to help better understand the correlation between flows and our later approximation method.

**Theorem 1.** Coding opportunity matrix \( P \) has the following properties:

1. Completeness property: \( \sum_{j=0}^{n-1} P_j = 1 \) (1 \( \leq i \leq n, 0 \leq j \leq n - 1 \).
2. Constraint on zero components: If \( z(j) \) is the number of zero components in vector \( \mathbf{C} \), then \( \max(z(j)) = n - (j+1) \). Similarly, if \( z(j) \) is the number of nonzero components in vector \( \mathbf{C}^i \), then \( \min(z(j)) = j + 1 \).
3. Equilibrium property: If \( z(j) = n - (j + 1) \) in vector \( \mathbf{C}^i \) for any two non-zero components, \( P_j \) and \( P_{j+1} \), we have

\[
\lambda_i \cdot P_j = \lambda_{j+1} \cdot P_{j+1}.
\]

**Proof 1.** Property (1) is obvious, because the events that a packet of flow \( i \) is \( j \)-level coded (\( j = 0, 1, \ldots, n-1 \)) are exclusive, and cover all possible processings of the packet.

To prove Property (2), assume that \( z(j) = n - (j + 1) \). We therefore have \( z(j) < j + 1 \), which means that at most \( j \) packets can be coded together. This is contradictory to the meaning of \( \mathbf{C}^i \), since a \( j \)-level coded packet means that there are \( j + 1 \) packets coded together. So, \( \max(z(j)) = n - (j + 1) \) and \( \min(z(j)) = j + 1 \).

Regarding Property (3), if \( z(j) = n - (j + 1) \), then \( z(j) = j + 1 \). Since the components in \( \mathbf{C} \) represent the probability that \( j + 1 \) packets are coded together. This requires the \( j + 1 \) flows each to contribute one packet to the coding. The limiting form of this situation is that the product of the mean arrival rate of flow \( i \) and \( P_j \) must be equal to the product of the mean arrival rate of flow \( k \) and \( P_{j+1} \), when \( P_j \) and \( P_{j+1} \) are both non-zero. □

4.2. The Calculation of \( E_\varphi \)

The exact calculation of \( E_\varphi \) is very hard. To avoid this difficulty, we assume that \( E_\varphi \) is independent of \( i \). This approximation is based on the intuition that from the viewpoint of any tagged queue (except queue \( j \)), the probability that queue \( j \) is empty should be roughly the same, since flow arrivals are independent. Therefore in the later discussion, we replace \( E_\varphi \) with \( E_\varphi \), which means the steady state probability that queue \( j \) is empty. We would like to point out that this approximation does not imply the mean arrival rate of flows is the same.

**Theorem 2.** For any \( 0 \leq i \leq n \),

\[
E_i = \frac{1 - \lambda_i \cdot g}{1 - \hat{F}_i},
\]

where

\[
0 \leq g < \min \left\{ \frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n} \right\}.
\]

**Proof 2.** Let \( \mathcal{F}_i \) stand for \( 1 - \hat{F}_i \), i.e., the probability that no packet will arrive to flow \( i \) within \( D \). According to (2), it is easy to translate \( P_i^{n-1} \) as:

\[
P_i^{n-1} = \prod_{k=1}^{\min\{z(j)\}} \left( 1 - E_k \cdot \mathcal{F}_k \right).
\]

Based on Eq. (8) and the properties of \( P \), we have for any \( i, k (1 \leq i, k \leq n) \),

\[
\frac{(1 - E_i \mathcal{F}_i)}{\lambda_i} = \frac{(1 - E_k \mathcal{F}_k)}{\lambda_k}.
\]

In order to decrease the number of unknown variables, we introduce a new parameter \( g \) into Eq. (9), then we obtain that for any \( i \)

\[
1 - E_i \mathcal{F}_i = \lambda_i \cdot g.
\]

We could ignore the case when \( \mathcal{F}_i = 0 \), because for a stochastic arrival process, the probability that no packets arrive within \( D>0 \), no matter how small, is not zero. We thus have

\[
E_i = \frac{1 - \lambda_i \cdot g}{1 - \hat{F}_i} \leq 1 - \lambda_i \cdot g.
\]

We could also ignore the case when \( E_i = 0 \), because \( E_i \) is the steady state probability that queue \( i \) is empty and should be non zero.

Since \( E_i \) and \( \mathcal{F}_i \) are probability values and thus are no larger than 1, from (10) we have \( 0 \leq \lambda_i \cdot g \leq 1 \), and hence \( 0 \leq g \leq \frac{1}{\lambda_i} \). Thus Equality (7) holds naturally. □

**Theorem 2** implies that the degree of freedom of the \( n \) variables, \( E_i (1 \leq i \leq n) \), turns out to be 1. To decide \( E_i \) we only have to focus on the value \( g \), whose calculation is to be disclosed in the next section.

4.3. Approximation with queue decomposition

Queue decomposition has been broadly used before, e.g., in [21], to tackle the difficulties in queueing analysis when multiple...
queues are entangled with interdependence. Based on the coding opportunity matrix $P$, we propose a new queue decomposition method to obtain the approximate solution to the performance of asynchronous network coding. Our method includes the following three steps:

- Step 1: Decompose the original queuing model (Fig. 2) into $n$ separate queues. Each queue $i$ ($i=1, \ldots, n$) has its own service model, called sub-system $i$ (Fig. 3).
- Step 2: Analyze sub-system $i$ ($i=1, \ldots, n$).
- Step 3: Obtain the approximate solution of the original queuing model by integrating the analytical results in Step 2.

4.3.1. Step 1

Decompose the original queuing model of asynchronous network coding (Fig. 2) into $n$ separate $GI/G/1/\infty$ queues, with each served by a sub-system. The input flow to queue $i$ ($i=1, \ldots, n$) remains the same as $A_i(t)$. Clearly, in order to analyze each queue separately, we need special treatment on its service, which is introduced in Step 2.

4.3.2. Step 2

To analyze the sub-system $i$ ($i=1, \ldots, n$), we treat the opportunistic delay as a kind of service, which is modeled by a virtual server. As such, the sub-system $i$ could be further decomposed by a virtual server followed by the actual service (i.e., coding and transmission), as shown in Fig. 3. Therefore, the total service time of the sub-system includes the "service" time of the virtual server plus the time of the actual service.

To obtain the "service" time distribution of the virtual server, we find that the row $i$ of $P$ illustrates the dependency between flow $i$ and other flows. Specifically, the value of $P_{ij}$ indicates the probability that a packet in flow $i$ is $j$-level coded. According to the rules of asynchronous network coding, the packet needs to wait for the maximum opportunistic delay $D$ before being actually served when $j < n - 1$, and it waits for a random opportunistic delay no larger than $D$ when $j = n - 1$. Therefore, the virtual server of sub-system $i$ can be further decomposed into 2 exclusive, parallel components (Fig. 3), with the probability to each component as $1 - P_i^{n-1}$ and $P_i^{n-1}$, respectively.

As shown in Fig. 3, the upper component of the virtual server has the entry probability $1 - P_i^{n-1}$ and has a deterministic "service" time $D$. The bottom component of the virtual server has the entry probability $P_i^{n-1}$ and has a random opportunistic "service" time no larger than $D$. Denote its distribution function and pdf as $S_i(t)$ and $s_i(t)$, respectively. The value of $S_i(t)$ is the probability that a packet of flow $i$ is $(n-1)$-level coded within the time period of $t$ conditioning on that the bottom component is used.

Using Eq. (2) and Theorem 2, we have

$$S_i(t) = \frac{\prod_{k=1} P_{ik}^{n-1} \cdot (1 - P_i(t))}{\prod_{k=1} P_{ik}^{n-1} \cdot g_{i}^{n-1}},$$

where the denominator is the probability that packets go into the bottom component (i.e., $P_i^{n-1}$) of the virtual server, and the numerator is the probability that the packets of flow $i$ is $(n-1)$-level coded within the time period of $t$. To simplify notations, we slightly abuse the notation of $F_i(t)$ by using $F_i(t)$ to denote the probability that there are packet arrivals to flow $i$ within the period of $t$. It is clear that $S_i(t) = 1$ when $t \geq D$.

Let $D_i(t)$ denote the distribution function of the "service" time of the virtual server in sub-system $i$. We have

$$D_i(t) = \begin{cases} P_i^{n-1} \cdot S_i(t), & t < D, \\ (1 - P_i^{n-1}) + P_i^{n-1} \cdot S_i(t) = 1, & t \geq D. \end{cases}$$

Let $r_i$ denote the total service time of sub-system $i$. Let $R_i(t)$ be the distribution function of $r_i$. Then $R_i(t)$ can be calculated as the convolution integral between the "service" time of the virtual server and the actual service time:

$$R_i(t) = \text{Prob}(d_i + x \leq t) = \int_0^t D_i(t-y) dB(y)$$

where $d_i$ and $x$ denote the "service" time of the virtual server and the actual service time.

From the above results, it is easy to see that $E_i$, $P_i$, $S_i(t)$, and $R_i(t)$ all are a function of $g$. Let $F_i^r$ denote the expectation of $r_i$. Then $F_i^r$ is also a function of $g$. Since $E_i$ is the steady state probability that queue $i$ is empty, based on the steady state relationship in queuing theory [36] and with Eq. (6) in Theorem 2, we can obtain another equation on $g$:

$$E_i = \frac{1 - \frac{\lambda_i}{g_i}}{1 - F_i^r} = 1 - \rho_i = 1 - \lambda_i F_i^r,$$

where $\rho_i$ is the utilization factor of subsystem $i$. Solving (15), we obtain the value of $g$.

4.3.3. Step 3

We can then obtain the approximate solution to the original queuing model, by integrating the analytical results in Step 2. We use the calculation of the average delay as an example:

(a) Choose any sub-system, say, sub-system 1. Use (6) and (8) to express $E_1$ and $P_j(0 \leq j \leq n - 1)$ as a function of $g$, respectively.
(b) Express $R_1(t)$ with (14). Calculate $F_i^r$, which is a function of $g$.
(c) Solve (15) to obtain the value of $g$, which will be used in the rest of calculations.
(d) For each sub-system, say, sub-system $i$, calculate the values of $E_i$ and $R_i(t)$ with (6) and (14), respectively. Calculate $F_i^r$, and Var($r_i$).
(e) For each $i = 1, \ldots, n$, calculate the variance of the inter-arrival times of flow $i$, denoted by $Var(a_i)$. Based on [36], the distribution of the idle-period, $I_a$ is the same as the residual time.

Fig. 3. The queuing model to approximate the processing of flow $i$ within a sub-system.
distribution of the input flow \( i \). The first two moments of \( i \), denoted by \( T_i \) and \( \bar{T}_i \), respectively, can be calculated from \( \bar{T}_i(t) \).

(f) The mean queueing time of flow \( i \) before going into the virtual server, \( \bar{T}_i \), can be calculated [36] as:

\[
\bar{T}_i = \frac{n^2}{2} \cdot (\text{Var}(a_i) + \text{Var}(r_i) + E_i^2) - \frac{\bar{T}_i}{2} \tag{16}
\]

The average delay of flow \( i \), \( T_i \) is thus:

\[
T_i = \bar{T}_i + \bar{T}_i \tag{17}
\]

5. Further discussion

In this section, we first discuss the condition that the queueing system is stable (i.e., no queue will eventually overflow). Then we analyze how to calculate the coding gain on certain network configurations.

5.1. Stability

Since our approximation method is based on the assumption that the queue size is infinite, according to the well-known Stability Theorem for GI/G/1/∞ [36], whether the steady state probabilities of interests (e.g., \( E_i \)) exist is decided by the stability of the queueing system. So, before applying our approximation method, we need to know under what conditions our queueing system is stable.

Theorem 3. The original queueing system (Fig. 2) is stable, if and only if in each sub-system \( i \), the maximum opportunistic delay \( D \), the mean arrival rate \( \lambda_i \) and the mean service rate \( \mu \) satisfy:

\[
D + \frac{1}{\mu} - P_i^{-1} \cdot \int_0^D S_i(t) dt < \frac{1}{\lambda_i} \tag{18}
\]

Proof. Clearly, the original queueing system is stable if and only if each sub-system is stable. In sub-system \( i \), the utilization factor \( \rho_i \) should be smaller than 1. Let \( \tau_i \) denote the expectation of \( r_i \) we first have:

\[
\tau_i = \left( 1 - P_i^{-1} \right) \cdot D + P_i^{-1} \cdot \int_0^\infty t \cdot s_i(t) dt + \frac{1}{\mu} = D + D \cdot P_i^{-1} + P_i^{-1} \cdot \left( \int_0^D t \cdot s_i(t) dt - \int_0^D S_i(t) dt \right) + \frac{1}{\mu} = D + D \cdot P_i^{-1} + D \cdot P_i^{-1} - P_i^{-1} \cdot \int_0^D S_i(t) dt + \frac{1}{\mu} = D + P_i^{-1} \cdot \int_0^D S_i(t) dt \tag{19}
\]

The theorem is proved since the utilization factor must be less than 1, i.e., \( \rho_i = \frac{\lambda_i}{\mu} > 1 \).

Theorem 3 implies that by solving Inequality (18), we can obtain a upper bound on the maximum opportunistic delay when other network configurations are fixed.

If the mean arrival rates of the \( n \) flows are not identical, we introduce a looser upper bound for the maximum opportunistic delay to quickly check the stability of the original queueing system. This could avoid intensive computation in solving Inequality (18).

Corollary 1. If the original queueing system is stable, the mean arrival rate \( \lambda_i \), the mean service rate \( \mu_i \) and the maximum opportunistic delay \( D \) should satisfy:

\[
D < \frac{1}{\lambda_{\text{max}}} - \frac{1}{\mu} \frac{1}{1 - \prod_{j=1}^{n} \lambda_j} \tag{20}
\]

where \( \lambda_{\text{max}} \) and \( \lambda_{\text{min}} \) denote the maximum mean arrival rate and the minimum mean arrival rate among the \( n \) flows, respectively.

Proof 4. Since \( S_i(t) \) is a distribution function, we have \( \int_0^D s_i(t) dt < D \). Thus Inequality (18) can be transformed as:

\[
\frac{1}{\lambda_i} > \frac{1}{\mu} - P_i^{-1} \cdot D \tag{21}
\]

From Theorem 2, since \( E_i \) cannot be zero and \( D \) is a bounded value in asynchronous network coding, we could ignore the case that \( P_i^{-1} = 1 \). Then we have

\[
D < \frac{1}{\mu} - \frac{1}{\lambda_i} \tag{22}
\]

Based on Theorem 2, we can obtain the upper bound of \( P_i^{-1} \), that is:

\[
P_i^{-1} = \prod_{j=1}^{n} \left( \frac{1}{\lambda_j} \min \left( \frac{1}{\lambda_j}, \frac{1}{\mu_j} \right) \right) \tag{23}
\]

If the original system is stable, Inequality (22) should hold for any sub-system. By considering the sub-system with the maximum mean arrival rate, we can obtain a looser bound on \( D \), which is equal to \( \frac{1}{\lambda_{\text{max}}} \).

Corollary 1 gives a necessary condition for system stability. It implies that if the differences between the mean arrival rates of the input flows are small and the maximum expected inter-arrival time \( \frac{1}{\lambda_{\text{max}}} \) is much larger than the mean service time \( \mu \), the maximum opportunistic delay could be set to a large value. Otherwise, it should be set small.

5.2. Coding gain

Coding gain is defined as the ratio of the number of transmissions required by forwarding, to the number of transmissions used by network coding to deliver the same set of packets. For simplicity, suppose that packets have the same size and delivering one (coded or uncoded) packet consumes the same bandwidth. The coding gain can be easily computed with the following theorem:

Theorem 4. Assume that the original queueing system is stable and the coding opportunity matrix is \( P \), the coding gain, \( G_c \), can be expressed as:

\[
G_c = \frac{\sum_{k=1}^{n} \lambda_k}{\sum_{k=1}^{n} \sum_{i=1}^{n} \left( \frac{n - 1}{n - 1} \right) \lambda_i \cdot P_i^{-1}} \tag{24}
\]

Proof 5. We need to compute both the expected number of total packets delivered per time unit and the expected number of coded packets delivered per time unit. Since the system is stable, the expected number of total packets delivered per time unit is equal to the expected number of packets arriving into the system per time unit, which is \( \sum_{i=1}^{n} \lambda_i \). With network coding, based on coding opportunity matrix \( P \), the expected number of coded packets with \( k \)-level coding opportunity per time unit is:

\[
\frac{1}{k+1} \sum_{i=1}^{n} \lambda_i \cdot P_i^k \tag{25}
\]

where \( \lambda_k \) means that the total number of delivered \( k \)-level coded packets has been counted \( k + 1 \) times. By summing up all
levels, we obtain the expected number of coded packets delivered per time unit as:

\[
\frac{1}{n} \sum_{i=1}^{n} \lambda_i \cdot P_i^{n-1} + \frac{1}{n-1} \sum_{i=1}^{n} \lambda_i \cdot P_i^{n-2} + \cdots + \frac{1}{1} \lambda_i \cdot P_i^n = \sum_{k=1}^{n} \sum_{i=1}^{n} \left( \frac{1}{n-k+1} \right) \cdot \lambda_i \cdot P_i^{n-k}.
\]

(26)

According to the above analysis and the meaning of coding gain, the coding gain can be calculated with Eq. (24). □

6. Model validation and performance evaluation

We have used a large amount of diverse network configurations to compare the performance (e.g. coding gain, average packet delay, and coding opportunity) calculated with our analytical model to that obtained with simulation. To save space, however, we only use three representative cases to illustrate the results, i.e., a simple case with memoryless traffic and service, a complex case with saturate non-memoryless traffic, and a more complex case with heavy-tailed traffic and adjustable service rate.

Both the numerical calculations and the simulations were implemented in Matlab. Since the network coding in the consideration of this paper only touches the network layer, we built an event-driven simulator to simulate the network layer operations and ignored the details of the physical layer and the MAC layer. Simulation results are obtained by taking average over 50 runs, each of which ends at 100,000 time units.

6.1. A simple case: memoryless traffic and service capacity

We first consider our approximation method in a simple case: the service time of the router follows an exponential distribution with the mean of 10 (time units), and that two input flows follow Poisson arrivals with the mean inter-arrival times of 15 and 20 (time units), respectively. With Theorem 3 and Corollary 1, the threshold value of \( D \) is equal to 13.8 (a tighter bound) and 20 (a looser bound), respectively. As shown in Fig. 4, when the system is stable, our analytical results are very close to those from the simulations. In Fig. 4, with the increase of \( D \), the coding gain, the average packet delay and the coding opportunity increase steadily. Under the constraint of a stable system, we may increase the maximum opportunistic delay to get a larger coding opportunity. This

![Fig. 4. Memoryless traffic and service capacity.](image-url)
operation brings a larger coding gain but also increases the average packet delay of the faster flow, as we can see from Fig. 4(b). See also, in Fig. 4(c), when $D$ approaches its threshold value, the 1-level coding opportunity of flow 2 is close to 1. At this situation, if we increase $D$ further, the system becomes unstable: the packets from flow 1 will be accumulated in the buffer, eventually resulting in buffer overflow no matter how large the buffer size is. This has been verified in our simulation.

6.2. A complex case: saturated traffic with balanced flow arrivals

Case 1 assumes Poisson traffic and exponential service. Our queueing model covers much beyond this simple case, and our analytical results hold for general distributions. To validate, we use a complex system configuration. Assume that the system has three independent input flows and their packet inter-arrival times all follow the Erlang-2 distribution with the same mean of 11 (time units). The service time of the router follows Gamma distribution with the shape parameter 2 and the scale parameter 1/5. The mean service time is derived from the Gamma-(2,1/5), which is equal to 10 (time units). Note that both Erlang and Gamma distributions have been broadly used in network traffic modeling [37].

This model has two features. First, the traffic load is nearly saturated, since the mean rate of each flow is very close to the service rate. Second, the flow arrivals are balanced, i.e., the mean arrival rates of flows are the same. We omit results of flow 2 and flow 3 in the figure, because the result is very similar to that of flow 1. As shown in Fig. 5(c), even if $D$ is set to be 0, the sum of 1-level and 2-level coding opportunities exceeds 0.95. With the increase of $D$, the 2-level coding opportunity of flow 1 increases steadily. With Theorem 3, the threshold value of $D$ in this case is about 4. From the figures, we can see that the analytical results match the simulation results nicely when the maximum opportunistic delay $D$ is smaller than its threshold. However, as in Fig. 5(b), after $D$ approaches and exceeds its threshold, the system becomes unstable and the average packet delays fluctuate severely. Our analytical average delays become meaningless.

6.3. A more complex case: heavy-tailed traffic with adjustable service rate

To further validate our analytical model for practical and more complex cases, we consider a system with heavy-tailed traffic arrivals of heterogeneous mean rates. It has been shown that the
network flows in Internet core may exhibit the self-similar nature, and their packet inter-arrival times could be approximated by a log-normal distribution [38], denoted by $\text{Log-N}(m, \sigma^2)$, where $m$ and $\sigma$ are the mean and the standard deviation of the inter-arrival time natural logarithm. Assume that there are three independent input flows following log-normal distribution with mean packet inter-arrival times of 20, 30, and 25, respectively. For simplicity, we set $m$ to 0, through calculation, the log-normal expression of these three distributions can be described as $\text{Log-N}(0,2.44772)$, $\text{Log-N}(0,2.60812)$, and $\text{Log-N}(0,2.53732)$, respectively. Assume that the service time of the router follows an exponential distribution with the parameter $\lambda$. We change $\lambda$ and the maximum opportunistic delay $D$ to study their impact on the performance.

As shown in Fig. 5, the analytical results match the simulation results very well when the mean service time and $D$ are both small. However, the gap between the two becomes a bit larger when the mean service time and $D$ increase. This is because a faster server (i.e., a smaller mean service time) and a smaller maximum opportunistic delay both imply less coding opportunities. With less coding opportunities, the multiple flows are less dependent on each other and our queuing decomposition method works better. As we can see in Fig. 6(b), (c), and (d), when $\mu$ and $D$ are set to 1 and 0, respectively, the coding opportunities for 2-level coding are all very close to zero and the average packet delay is also at its lowest point, as in Fig. 6(a). From Fig. 6(a), we can also see that the average packet delay is more sensitive to the service time than to the maximum opportunistic delay. This is because the service time has a direct impact on the packet delay, while the maximum opportunistic delay might not. Note that the maximum opportunistic delay is only an upper bound on the waiting time for better coding opportunities. Packets do not have to wait to this bound before being processed.

7. Self-adjustable delay-based coding mechanism

In this section, we illustrate how to use our analytical results to design an adaptive network coding mechanism for efficient congestion control.

Most traditional IP layer congestion control mechanisms use Active Queue Management (AQM) policies [39], such as BLUE [40], to keep queue length small. However, these policies share a common drawback that there is a time lag between the congestion notifications issued by the router to the end hosts’ rate control actions being in effect. During this lag, the congestion may become even worse, and more packets are dropped.
Network coding technique brings a new solution for IP congestion control [24]. As shown in Example 1, once the congestion is detected, the router activates network coding mechanism, and informs its end hosts to transmit original packets through redundant paths. With network coding, more than one buffered packets can be removed from the congested router simultaneously, which may significantly reduce the packet losses. However, to maximize the coding opportunities and the coding gains while controlling the congestion is not trivial. Since in real-world deployment, the traffic loads are time-varying, a fixed value of the maximum opportunistic coding opportunities and the coding gains while controlling the congestion is not sufficient. To address this problem, we design a Self-Adjustable Delay-based Coding Mechanism, called SADCM. By using our analytical results and the periodic network measurements, SADCM can quickly find out the near optimal values for the maximum opportunistic delay, which help to maintain high-level coding gains. To guarantee congestion control effectively, SADCM can also adjust the maximum opportunistic delay dynamically and sensitively as the input traffic loads are changing. The details of SADCM are described as follows:

- **Measurements collection**: Suppose that the congestion control operates periodically with equal length time intervals, each of which are set to $T$ time units. During each interval, the SADCM router measures arrival rates for the input flows and its service rate by using Exponential Weighted Moving Average (EWMA) approach [39]. For instance, flow $i$’s EWMA arrival rate at the $r$th interval, $\tilde{\lambda}_i(r)$, can be calculated as:

$$\tilde{\lambda}_i(r) = \alpha \lambda_i + (1 - \alpha)\tilde{\lambda}_i(r - 1),$$

where $\alpha$ is the weighting factor, and $\lambda_i$ is the measurement of the arrival rate at the $r$th interval. A higher $\alpha$ discounts older observations faster.

- **Congestion detection**: To evaluate congestion, the router needs to estimate the queue length threshold for each flow at the end of each interval. Let $\delta_i(r)$ denotes the threshold of the flow $i$ at the $r$th interval, Suppose that there are $n$ input flows, which can be estimated as:

$$\delta_i(r) = K(1 - P_{cv}) - T \cdot \left(\max_{i} \left(\frac{1}{\tilde{\lambda}_i(r)} - \frac{1}{\tilde{\lambda}_i} \cdot \tilde{\lambda}_i(r) \cdot \sum_{i=1}^{n} \tilde{\lambda}_i(r) \cdot 0\right)\right),$$

where $K$ is the maximum queue size and $P_{cv}$ is the reserved portion of the queue space for traffic bursty. If the aggregated EWMA arrival rate of the input flows is larger than the EWMA service rate, that is, $\sum_{i=1}^{n} \tilde{\lambda}_i(r) > \tilde{\lambda}(r)$, and if there is one queue, say $k$, its length $L_k$ exceeds $\delta_k(r)$, the router activates SADCM for congestion control.

- **Self-adjustable mechanism**: After being activated, SADCM first marks output packets as the congestion notifications. Then, by using Corollary 1, it calculates a region $[D_{\text{min}}, D_{\text{max}}]$ to bound the optimal value of $D$ at the recent time intervals. As shown in Algorithm 1, a binary search is used to quickly refine the region size to an error interval, denoted by $\epsilon$. As we can see from the algorithm, SADCM periodically counts the coding gain, denoted by $G_i(\eta)$, which is used to compute the service rate under network coding. If the aggregate EWMA arrival rate is smaller than the service rate under network coding, SADCM regards that the congestion is being alleviated. If the input traffic loads are changed or the router is fed by new input flows, SADCM will reset the binary search process. The pseudo-code of the self-adjustable mechanism is shown in Algorithm 1.

- **Congestion release**: At the beginning of another time interval, say $v$, if $\sum_{i=1}^{n} \tilde{\lambda}_i(v) < \tilde{\lambda}(v)$, and for any $i$, $L_i < \theta_i(v)$, the router switches SADCM back to the store-and-forward routing, which reduces the complexities brought by the coding operations.

To evaluate the performance of SADCM, we modify Case 1 in Section 6.1 as follows: assume that initially the router has only one input flow with the mean inter-arrival time of 15 (time units). After 10,000 time units, a flow with the mean inter-arrival time of 20 (time units) is fed into the router. At the simulation time of 40,000 time units, the second input flow with mean inter-arrival times of 20 finishes its transmission. The specific configuration of this case study is described in Table 2.

As shown in Fig. 7, at the time period between 10,000 and 10,500 time units, the router finds that the aggregated EWMA arrival rate becomes larger than the EWMA service rate, and both the queue lengths and the packet delays of the two input flows increase rapidly. At the beginning of next period (i.e., between 10500 and 11,000 time units), the queue length of flow 2 exceeds its threshold, the router thus activates SADCM for congestion control. As we can see from Fig. 7(a), the router can quickly find a tight region to bound the maximum opportunistic delay to approach the maximum coding gain. As shown in Fig. 7(c) and (b), with network coding, flow 2 maintains relatively stable queue length and stable packet delay, since newly-joined packets of flow 2 can be immediately coded with the queueing packets of flow 1. However, the opportunistic delay enlarges both the queue length and the packet delay of flow 1. Fig. 7(d) illustrates that SADCM maintains a high coding gain.

We also implement the BLUE [40] for comparison. The BLUE router maintains a probability $p_m$, which decides the chance of dropping or marking the packets when they arrive. If the packets are dropped due to overflow, the router increases $p_m$ by $\theta_1$; in contrast, if the downstream links are idle, the router decreases $p_m$ by $\theta_2$. Usually $\theta_1$ is much larger than $\theta_2$, BLUE also uses a parameter

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Case configuration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Settings</td>
</tr>
<tr>
<td>Mean arrival rate of flow 1 $(\lambda_1)$</td>
<td>1/15</td>
</tr>
<tr>
<td>Mean arrival rate of flow 2 $(\lambda_2)$</td>
<td>1/20</td>
</tr>
<tr>
<td>Maximum buffer size (K)</td>
<td>20</td>
</tr>
<tr>
<td>Mean service rate (µ)</td>
<td>1/10</td>
</tr>
<tr>
<td>Length of time period (T)</td>
<td>500</td>
</tr>
<tr>
<td>Additive increment of $D$ ($\Delta_0$)</td>
<td>0.2</td>
</tr>
<tr>
<td>Bound region size of $D$ ($\epsilon$)</td>
<td>1</td>
</tr>
<tr>
<td>EWMA weighting factor ($\alpha$)</td>
<td>0.8</td>
</tr>
<tr>
<td>Reserved portion of the queue space ($P_{cv}$)</td>
<td>1/4</td>
</tr>
</tbody>
</table>
to determine the minimum time interval between two successive updates of $p_m$. The pseudo-code of BLUE is shown in Algorithm 2, in which $T_{\text{now}}$ and $T_{\text{last}}$ denote the current time instant and the instant when $p_m$ was updated last time, respectively.

**Algorithm 2. BLUE AQM scheme**

Upon packet loss event:

\[
\text{if } (T_{\text{now}} - T_{\text{last}}) > T_{\text{freeze}} \text{ then} \\
p_m = p_m + \theta_1; \\
T_{\text{last}} = T_{\text{now}}.
\]

Upon link idle event:

\[
\text{if } (T_{\text{now}} - T_{\text{last}}) > T_{\text{freeze}} \text{ then} \\
p_m = p_m - \theta_2; \\
T_{\text{last}} = T_{\text{now}}.
\]

Both SADCM and BLUE are evaluated by two groups of configurations without explicit congestion notifications (ECN). The detailed configurations are listed in Tables 3 and 4, respectively. In our comparison, two input flows with the mean inter-arrival time of 15 and 20 (time units) feed their packets to the router during the time period between 0 and 10,000 time units. Two metrics, the packet loss rate and the average queue length, are recorded under various maximum queue sizes, increased from 20 to 100. As shown in Fig. 8(a), the packet loss rate of SADCM is much lower than that of BLUE, since SADCM adjusts the maximum opportunistic delay to control the queue length and encodes the packets of the two input flows together. The average queue sizes shown in Fig. 8(b) illustrate that to improve the coding gain, the maximum opportunistic delay in SADCM leads to a slight longer queue.

**Table 3**

<table>
<thead>
<tr>
<th>Configurations</th>
<th>SADCM-1</th>
<th>SADCM-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(T_f)$</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>$(\Delta_d)$</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$(\alpha)$</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>$(P_{\text{rv}})$</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

**Table 4**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>BLUE-1</th>
<th>BLUE-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{freeze}}$</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>$\theta_1$</td>
<td>0.02</td>
<td>0.005</td>
</tr>
<tr>
<td>$\theta_2$</td>
<td>0.002</td>
<td>0.0005</td>
</tr>
</tbody>
</table>
Overall, SADCM can reduce packet losses significantly and can achieve a high coding gain. What's more, it can switch easily between store-and-forward routing and the delay-based network coding mechanism. In this sense, SADCM is an important complementary component for recent AQM schemes.

8. Conclusion

While the research and development on network coding have remained active for nearly a decade, its queuing behavior in the Internet core are still far from clear. Much attention has been given in the study of network coding performance in practical systems, e.g., in [2,3,6]. Yet, due to the special difficulties in modeling the intricate correlation among different flows, quantitative analysis and evaluation on the interplay of key components of asynchronous network coding are blank, especially when the traffic flows and the service time are stochastic and may not have the Markov-type property. To fill this vacancy, we first develops a queuing model and solves the hard queuing problem with queue decomposition based on the approximation of flows’ coding opportunities. Then we study the sufficient and necessary condition for a network coding system to be stable and discuss what the coding gain is in a stable system. Based on our analytical results, we finally propose and a self-adjustable delay-based coding mechanism for effective congestion control. Our work on the relationship of network coding performance and the core system parameters can help better protocol design and system configuration.

Acknowledgments

The work was supported by the National Basic Research Program of China (2011CB302601) and the National High Technology Research and Development Program of China (2012AA01A301) and General Research Fund of the Hong Kong SAR, China No. (CityU 114609) and CityU Applied R & D Funding (ARD) Nos. 9681001, 6351006 and CityU Strategic Research Grant No. 7008110.

References


