

An Efficient Random Routing Transmission Mechanism with Raptor Coding and Adaptive Redundancy Control

Yanyan Xue* **, Jiali You* **, Jun Chen* **

* National Network New Media Engineering Research Center, Institute of Acoustics,
Chinese Academy of Sciences, Beijing, China

** School of Electronic, Electrical and Communication Engineering,
University of Chinese Academy of Sciences, Beijing, China

xueyv@dsp.ac.cn, youjl@dsp.ac.cn chenj@dsp.ac.cn

Abstract— Random-routing systems such as KRRD often suffer from long-tail latency and elevated packet-loss rates due to unpredictable routing paths. To address these challenges, we design an efficient random-routing transmission mechanism built upon rateless Raptor coding. By leveraging its strong end-to-end forward error-correction (FEC) capability, the proposed mechanism effectively mitigates performance degradation caused by long-tail effects (excessive session completion time). To further improve transmission efficiency, we introduce an adaptive redundancy regulation scheme driven by decoding-state feedback, referred to as ARRCT. The receiver dynamically reports redundancy requirements according to real-time decoding progress and the observed packet-loss rate, enabling the sender to inject additional coded packets on demand. This closed-loop adaptation strikes a balance between transmission reliability and bandwidth utilization. ns-3 simulation results show that, for packet-loss rates between 0.01% and 1.0%, ARRCT reduces session completion latency by approximately 10%–20% compared with baseline random-routing schemes. Under severe loss conditions, ARRCT maintains a high decoding probability and significantly enhances the robustness of random-routing covert communication systems.

Keywords—Adaptive redundancy, Forward error correction, KRRD, Random routing, Raptor code

I. INTRODUCTION

With the continuous development and evolution of network technologies, the functionality of various network services has become increasingly diverse, but the accompanying security threats have also grown more severe. Traditional secure communication mainly relies on encryption to protect the content of communications [1]; however, attackers can still identify high-frequency encrypted links through traffic analysis and feature recognition, and then launch targeted interference, tampering, or blocking attacks against the communication paths, thereby degrading communication covertness and undermining reliability. Covert communication, by contrast, leverages random routing and traffic dispersion to effectively hide the relationship between the communicating parties and the

characteristics of the communication paths [2]. For example, the Crowds system [3] achieves anonymous forwarding by allowing nodes to randomly select the next hop; it offers good scalability and relatively low latency, but its covertness is limited. Systems such as Vuvuzela [4], Riposte [5], and Tor [6] provide strong covertness, but their transmission performance is constrained: they typically incur high latency and are not well suited for large-volume data transfer. Existing systems usually hide packet metadata, but they operate at the flow granularity and do not conceal traffic behavior. When a group of packets is transmitted along the same underlying path, the resulting traffic pattern becomes easily observable and attackable, thereby compromising covertness. To enhance resistance against traffic-analysis attacks, several packet-granularity systems have been proposed in which each packet follows an independent transmission path, thereby mitigating traffic-behavior features. Representative systems include Loopix [7], draughts [8], and KRRD [2]. Loopix follows the mix-net paradigm [9] and depends on global information, which is unfavorable for decentralization and horizontal scalability; draughts and KRRD, in contrast, follow the random routing paradigm. Draughts selects the next hop and the next-next hop based solely on local information within two hops, without relying on a global view, while KRRD selects the next hop using only one-hop local information. However, neither of these systems considers the issues that arise in high-packet-loss transmission scenarios (e.g., TCP-based communication).

Under a random routing forwarding mechanism, the next hop of each packet is independently and randomly selected by intermediate nodes, leading to significant variations in path length and link quality across different packets. This gives rise to a typical “long-tail effect”: some packets experience abnormally long transmission times or even time out and are lost, which in turn increases the overall session delay. In addition, the high uncertainty of random paths makes it difficult for traditional ACK/retransmission-based in-order delivery mechanisms to effectively cope with severe packet loss and reordering.

Raptor coding [10], as a rateless fountain code, provides strong resilience to packet loss and features low encoding/decoding complexity, enabling efficient data recovery without relying on retransmissions. Therefore, this paper attempts to incorporate a Raptor coding mechanism into random routing transport systems to address the high loss rate and long-tail effect caused by random forwarding. Meanwhile, to avoid bandwidth waste and other efficiency issues induced by excessive redundancy, we design a feedback-controlled dynamic redundancy mechanism to achieve a self-balancing trade-off between coding overhead and transport performance. The main contributions of this paper can be summarized in the following two aspects:

Building on KRRD, we design a novel transport scheme that integrates forward error correction with random routing forwarding. By introducing Raptor coding, the scheme enhances the reliability and robustness of random-walk networks, and we further propose an adaptive redundancy control algorithm (ARRCT) based on decoding feedback. Through a closed-loop feedback mechanism, the receiver dynamically requests additional redundancy, while the sender generates and transmits encoded packets on demand, thereby achieving a joint optimization of reliability and transmission efficiency.

We implement an end-to-end simulation platform in ns-3 and conduct comparative experiments to evaluate the proposed scheme under different packet loss rates, redundancy ratios, and traffic loads. The results show that, while preserving path covertness and traffic balance, the ARRCT mechanism reduces transmission delay by approximately 10%–20% compared with traditional random routing transport schemes when the packet loss rate ranges from 0.02% to 0.10%.

II. RELATED WORK

Random routing [11] is a representative technique in covert communication systems. Existing work can be broadly categorized into two classes: methods based on random routing and path dispersion, and methods based on traffic mixing and mix networks. Systems that realize covert transmission via random routing and path dispersion include Crowds and KRRD (K-randomized routing dispersion). The Crowds system uses a preset forwarding probability p_f to choose between “continue random forwarding” and “send directly to the destination”. This design is structurally simple, incurs low forwarding overhead, and performs well in terms of throughput and delay. However, its path selection is stateless and independent across hops, and it lacks traffic-mixing mechanisms such as delay mixing and batching, which makes it vulnerable to long-term traffic correlation attacks and limits its level of covertness. KRRD is a recently proposed packet-granularity random forwarding overlay architecture. It combines link-independence-aware neighbor selection with a load-aware set of K candidates to spatially disperse traffic and avoid eavesdropping fate sharing (EFS). In addition, by leveraging ICN name-based addressing together with per-hop address rewriting, KRRD hides endpoint relationships at the addressing layer. The KRRD architecture typically couples a distributed control plane with a fast data

plane to enable “on-arrival” random forwarding decisions and path anonymity, and is representative in terms of anti-eavesdropping capability and scalability.

Systems based on traffic mixing and mix networks include Vuvuzela [4] and Loopix [7]. Vuvuzela injects large volumes of noise messages (cover traffic) between servers to globally obscure real communication relationships, providing very strong traffic confidentiality and formal security guarantees under a powerful global passive adversary. However, this “high-intensity cover traffic” brute-force mixing is extremely costly: the large amount of noise consumes bandwidth and processing resources, resulting in very low effective throughput and very high end-to-end latency, which makes it difficult to support large-scale or high-frequency communication scenarios. Loopix further systematizes the combination of random-delay queues, a Poisson process model for message arrivals, and continuous cover traffic. By randomizing timing, traffic volume, and paths in multiple dimensions, it effectively hides traffic patterns and offers strong, provable resistance to traffic analysis. Nevertheless, such mix networks inherently introduce random delays and rely on continuous cover traffic to maintain security, leading to a significant drop in transmission efficiency: median link latency increases, jitter is aggravated, and overall throughput remains only moderate to low.

To address the performance limitations of existing random routing transmission systems, we use KRRD as the baseline framework and study transmission optimization under its random forwarding paradigm and security constraints, targeting two key issues: long-tail delay under high packet loss and large latency induced by uncertain routing paths. Our design philosophy is to couple a reliable coding mechanism with random routing. For clarity, the key symbols used in this paper are summarized in Table 1. Among candidate FEC [12] schemes, Raptor coding—a typical fountain code—naturally fits random routing, high-loss environments. In Raptor coding, the original data is first divided into k fixed-size source packets. Pre-coding linearly combines the k source packets into n intermediate packets. These intermediate packets represent a data state between the source packets and the final encoded packets, and are used as the input to the subsequent LT encoding stage. On top of this, the encoder employs an LT-type fountain mechanism [13] to generate encoded packets on demand: for each encoded packet, it samples a degree d from a predefined degree distribution, randomly selects d intermediate packets, and computes their bitwise XOR to obtain one encoded packet. By repeating this process, the encoder can generate an arbitrary number of encoded packets online without fixing the code rate in advance. At the receiver side, it is sufficient to gather a collection of encoded packets that is marginally larger than k and that satisfies certain dependency constraints in order to reconstruct all k source packets. The need for more than k packets is inherent to Raptor code design: a minimum decodable redundancy ε_k [14][15] is required to fully recover the source data. If we define s_t as the effective arrival ratio of encoded packets, then, for a static network, the redundancy ratio r has the provable lower bound $r \gtrsim \frac{1+\varepsilon_k}{s_t} - 1$. This combination of on-demand redundancy generation and

successful decoding from only slightly more than k received packets makes Raptor coding especially well matched to environments with time-varying link loss that is hard to estimate accurately, random multi-hop overlay topologies, and limited state information about communicating endpoints.

TABLE 1. SYMBOL DEFINITIONS

Table 1. Notation and Symbol Definitions	
Symbol	Definition
r	Encoding redundancy ratio
k	Number of source packets
ε_k	Minimum redundancy overhead required for successful decoding
s_t	Estimated effective arrival probability, $s_t = N_t/n_t$
n_t	Total number of encoded packets
N_t	Number of encoded packets actually received
τ	Decoding timing window

III. SYSTEM ARCHITECTURE OF ARRCT

As shown in Fig. 1, the system architecture integrates random routing with Raptor coding and consists of four core modules. The encoding module at the sender divides the original data block into k packets and generates encoded packets for transmission. The random routing transmission module performs hop-by-hop routing decisions at intermediate nodes. On the receiver side, the decoding module buffers incoming coded packets, performs decoding, and manages the decoding cycle. Finally, the feedback regulation module dynamically updates the redundancy ratio based on the computed effective arrival rate, enabling adaptive system adjustment.

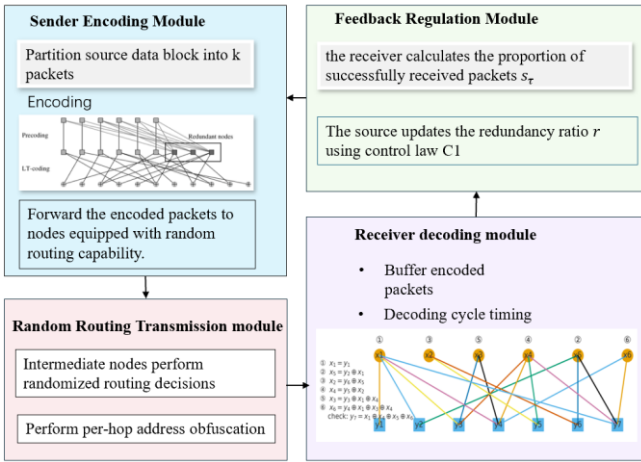


Figure 1. Framework of the ARRCT random routing transmission system.

A. Dynamic Redundancy Adaptation Based on Decoding and Real-Time Network Feedback

The core idea of ARRCT is to achieve long-tail latency suppression by “trading redundancy for waiting” under the KRRD [7] random routing model, while dynamically adapting the Raptor coding redundancy at the sender subject to successful decoding. In a static network, link capacity is

assumed to be stable; with a fixed redundancy ratio, Raptor coding is performed at the sender, and the receiver can decode and recover all k source packets once it has collected slightly more than k encoded packets. In contrast, in a random routing network, packet paths are neither fixed nor predictable, and the instantaneous transport capacity of the network also varies over time. As a result, the fraction of encoded packets that successfully reach the receiver fluctuates: when the network is in good condition or a coded packet happens to traverse a relatively good path, the number of packets arriving at the receiver is sufficient; when the network is degraded or packets are forwarded along poorer paths, the number of arrivals may be insufficient to reach the decoding threshold. ARRCT is designed precisely to address decoding failures in such dynamic environments, where static redundancy cannot accommodate the unpredictable random-walk paths and time-varying network conditions.

In the proposed mechanism, the k source packets are pre-coded and LT-encoded at the sender to produce n encoded packets, where $n = k(1 + r)$. The sender then injects these packets into the network, where each packet undergoes random routing—being forwarded to the next relay with probability P_f or sent directly to the receiver with probability $1 - P_f$. At the receiver, decoding is attempted while simultaneously counting the actual number of arrived encoded packets, which is used to compute the effective arrival probability s_t of the encoded packets. After the decoding window τ ends, s_t is calculated and the failure flag $fail_t$ is set: if decoding completes within the window τ , then $fail_t = 0$; otherwise, if decoding times out, $fail_t = 1$. At the end of each window, the receiver reports these decoding statistics back to the sender to update the redundancy r ; the feedback is routed along random paths and encapsulated with fixed-length padding to obfuscate its semantics.

If the redundancy ratio were adjusted solely based on the decoding status ($fail_t$), it would disregard both the instantaneous and historical network conditions that shape the redundancy demand. Therefore, we propose a control law that jointly accounts for these factors, as expressed in the following equation. As shown in this equation, the three multiplicative terms on the right-hand side correspond, respectively, to the redundancy demand under historical network conditions, the instantaneous lower bound on the redundancy required by the current network state, and the additional redundancy needed to compensate for decoding failures. This control law employs a projection operator $\Pi_{[r_{\min}, r_{\max}]}$ to map the computed result onto a feasible redundancy interval $[r_{\min}, r_{\max}]$, which not only ensures engineering practicality but also guarantees rapid recovery in the event of decoding failures. Physically, the penalty term $kfail_t$ ensures that the system does not remain trapped for a long time in an “under-provisioned redundancy” regime; that is, it serves as a safety valve that prevents persistent under-allocation of redundancy. In this mechanism, the packet handling process is illustrated in Fig. 2, where each small rectangle denotes a packet and gray rectangles represent lost or timed-out packets. As shown in the figure, the core idea of the mechanism is to compensate for timeout losses by injecting additional redundancy.

$$r_{t+1} = \Pi_{[r_{\min}, r_{\max}]} \left(\lambda r_t + (1 - \lambda) \left(\frac{1 + \varepsilon_k}{s_t} - 1 \right) + \kappa \text{fail}_t \right).$$

where

- r_{t+1} is the redundancy ratio used for encoding in round $t + 1$;
- r_t is the redundancy ratio used for encoding in round t ;
- N_t is the number of encoded packets actually received in round t ;
- n_t is the total number of encoded packets generated in round t ;
- $s_t = N_t/n_t$ is the estimated effective arrival probability observed in round t ;
- fail_t is the decoding-failure indicator in round t ($\text{fail}_t = 1$ if decoding fails in round t , and $\text{fail}_t = 0$ otherwise);
- $\lambda \in [0,1)$ is the forgetting factor (smoothing weight). A larger λ (closer to 1) assigns more weight to historical values, making r smoother but less responsive; a smaller λ makes r rely more on the latest observation, leading to faster response but larger fluctuations. This acts as an inertia/smoothing term on the control variable, effectively adding damping to the redundancy ratio r and preventing violent oscillations caused by instantaneous randomness in the random routing process. This parameter is set empirically;
- r_{t+1}^{raw} is the input to the projection operator, $r_{t+1}^{\text{raw}} = \lambda r_t + (1 - \lambda) \left(\frac{1 + \varepsilon_k}{s_t} - 1 \right) + \kappa \text{fail}_t$.
- $\Pi_{[r_{\min}, r_{\max}]}$ is the projection operator that maps the updated redundancy ratio onto the feasible interval $[r_{\min}, r_{\max}]$. $r_{\min}$ is the minimum value of the dynamically adjusted redundancy ratio, which is typically set to around 0.05 in practice. The upper bound $r_{\max}$ for dynamic redundancy adjustment can be chosen according to the simulation setup or practical deployment constraints; in our simulations, it is set to 0.5.

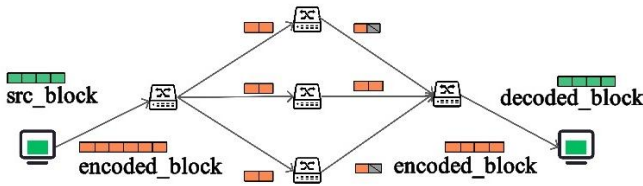


Figure 2. ARRCT random routing data block processing flow.

B. ARRCT Algorithm Design

The ARRCT mechanism mainly comprises the receiver-side feedback procedure and the sender-side redundancy update procedure, as shown in Table 2.

Algorithm 1: Feedback-Driven Redundancy Update

Input: threshold ε_k ; timeout τ ; current redundancy ratio r_t ; redundancy bounds $[r_{\min}, r_{\max}]$; control parameters λ ;

Output: Estimated success rate s_t ; failure indicator fail_t ; updated redundancy ratio r_{t+1} .

Receiver-Side Feedback Procedure

- 1: Initialize the counter of successfully received packets: $N_t \leftarrow 0$.
- 2: Within the timeout τ , count all valid received packets and update N_t .
- 3: Compute the empirical success rate: $s_t \leftarrow N_t / n_t$.
- 4: If $N_t < k(1 + \varepsilon_k)$, set $\text{fail}_t \leftarrow 1$; otherwise, set $\text{fail}_t \leftarrow 0$.
- 5: Construct a feedback message containing (s_t, fail_t) .
- 6: Send the feedback message to the sender via random return paths.

Sender-Side Redundancy Update Procedure

- 1: Extract s_t and fail_t from the received feedback.
- 2: Compute raw redundancy update:
$$r_{t+1}^{\text{raw}} \leftarrow \lambda r_t + (1 - \lambda) \left(\frac{1 + \varepsilon_k}{s_t} - 1 \right) + \kappa \text{fail}_t$$
- 3: Project r_{t+1}^{raw} onto $\Pi_{[r_{\min}, r_{\max}]}$ to obtain r_{t+1} .
- 4: Use r_{t+1} for the next transmission round.

IV. EXPERIMENT

To evaluate the long-tail suppression and loss-resilience capabilities of the proposed Raptor-coded ARRCT mechanism in decentralized random routing covert transmission, we built an ns-3 simulation platform on a Dell PowerEdge R720 server running Linux (Ubuntu 20.04.6 LTS, Intel Xeon E5-2603 v2 @ 1.80 GHz).

A. Experiment Settings

To clearly observe per-packet behavior and routing paths, we adopt the simple topology shown in Fig. 2 and implement KRRD-based random routing transmission. All links are point-to-point, configured with a data rate of 10 Mbps and a propagation delay of 2 ms. The experiment compares three transmission schemes:

- Uncoded KRRD random routing covert transmission
- Raptor-coded KRRD random routing covert transmission with a fixed redundancy ratio.
- ARRCT-based KRRD random routing covert transmission with adaptive redundancy feedback.

B. Experiment Results and Analysis

The first experiment aims to verify the loss-tolerance capability of Raptor coding by comparing uncoded random routing covert transmission against its Raptor-coded version. Results indicate that Raptor coding yields a notable reduction in session completion time, achieving a 10%–20% improvement. As depicted in Fig. 3, for the same source-data size, the Raptor-coded random routing scheme (blue dashed line) exhibits significantly shorter session duration than the uncoded approach (red solid line). These findings demonstrate that Raptor coding can effectively mitigate long-tail delays caused by packet drops and timeout retransmissions in randomized forwarding environments.

To further analyze the impact of the redundancy ratio on transmission performance, we conduct a second set of simulations to evaluate the total transmission delay under different redundancy ratios and packet loss rates. The results are shown in Fig. 4. For a fixed packet loss rate, when the redundancy ratio is smaller, the overall transmission delay is lower. Fig. 5 (the surface of session completion time versus packet loss rate and source data size under different redundancy ratios) further demonstrates that, beyond the minimum redundancy required to ensure successful decoding, increasing the redundancy ratio still leads to a growth in total transmission delay. The reason is that, under a fixed-rate coding scheme, a larger redundancy ratio r implies a larger number of encoded packets n to be sent, which in turn increases the sender's queueing delay, link occupation time, network congestion level, and the waiting time at the receiver before decoding can start. Therefore, although Raptor coding can significantly reduce end-to-end delay, the redundancy level must be carefully controlled so as not to introduce unnecessary bandwidth overhead and transmission latency that would offset the performance gains brought by coding.

The third experiment evaluates the proposed ARRCT adaptive redundancy mechanism, motivated by the performance limitations observed in fixed-redundancy schemes. To achieve dynamic matching between redundancy and network conditions, we compare ARRCT against fixed-redundancy Raptor coding. As shown in Fig. 6, ARRCT consistently outperforms the static scheme, clearly demonstrating the benefit of adaptive feedback. To further quantify this improvement, we plot the speedup curve (Speedup = fixed-redundancy session time / adaptive-redundancy session time), shown in Fig. 7. The results reveal several trends:

Rising trend: Speedup increases as the source-data size grows, indicating that ARRCT provides more substantial benefits for large-scale transmissions.

Small data sizes: For small data blocks (1–10 MB), the performance gap between the two schemes is minor, and ARRCT even dips slightly below 1 at some points. This is due to additional state evaluation and feedback overhead, which becomes noticeable when the total transmission time is short.

Large-scale advantage: As data size increases, ARRCT's advantage rapidly expands. Beyond 20 MB, speedup consistently exceeds 1, demonstrating that fixed redundancy can no longer match network conditions, leading to avoidable overhead.

Peak performance: In the 40–50 MB range, speedup peaks at approximately 1.6×, meaning ARRCT reduces session completion time by about 50% under the same network conditions.

Overall, the speedup curve demonstrates the effectiveness of ARRCT: by dynamically adjusting redundancy based on real-time network conditions and data volume, it eliminates the inefficiency of static redundancy and achieves lower session completion time and higher transmission efficiency.

C. Eavesdropping and Decryption Resistance Evaluation

ARRCT provides end-to-end resistance to data compromise through Raptor coding. We define the eavesdropping ratio ρ_{eave} as the proportion of encoded packets captured by the adversary, and the deciphering probability $P_{decipher}$ as the probability that the adversary can fully reconstruct the source data.

Without coding, $P_{decipher} \approx \rho_{eave}$ (Fig. 8, blue). With Raptor coding, however, decoding succeeds only when the adversary collects more than the required number of packets, i.e., $N_{eave} \geq K(1 + \epsilon_k)$.

Expressing redundancy as: $R_{code} = \frac{K}{N} = \frac{1}{1+r}$ the red curve in Fig. 7 (example: $r = 0.25 \rightarrow R_{code} = 0.8$) shows that:

For $\rho_{eave} < R_{code}$, $P_{decipher} \approx 0$;

For $\rho_{eave} \geq R_{code}$, $P_{decipher} \approx 1$.

Thus, ARRCT increases the critical threshold for successful data recovery. Furthermore, under KRRD's randomized path dispersion, an adversary must compromise a significant fraction of nodes to raise ρ_{eave} . Based on KRRD [11], achieving $P_{decipher} > 0.8$ often requires compromising more than 25% of the nodes—already rendering the network nearly unusable. Therefore, ARRCT delivers performance improvements without reducing the system's inherent security.

D. Section Summary

The experimental results clearly demonstrate that ARRCT provides significant performance advantages over both uncoded and fixed-redundancy schemes in decentralized random routing covert transmission. **Reliability:** Raptor coding substantially enhances loss resilience and effectively suppresses long-tail delay by trading redundancy for waiting time. **Efficiency:** ARRCT adaptively adjusts redundancy to balance transmission reliability and bandwidth utilization, avoiding the overhead inherent in fixed-redundancy designs. **Practical Value:** In large-scale data-transfer scenarios, ARRCT achieves a speedup of up to 1.6×, highlighting its strong potential for high-performance covert transmission over random routing networks.

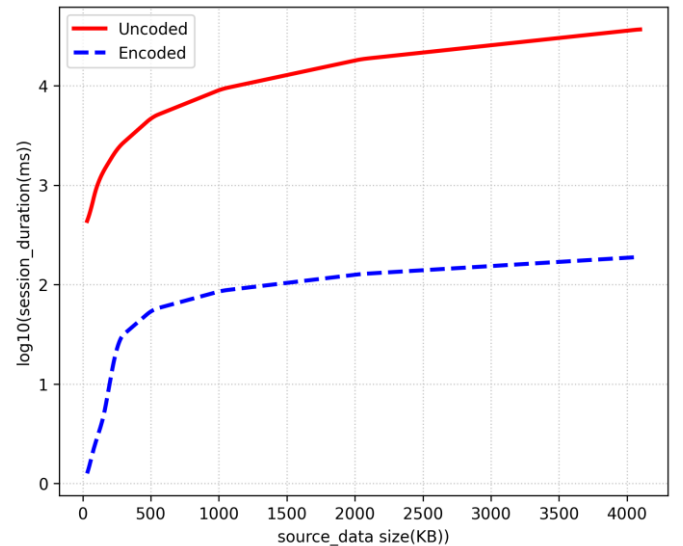


Figure 3. Session completion time vs. source data size

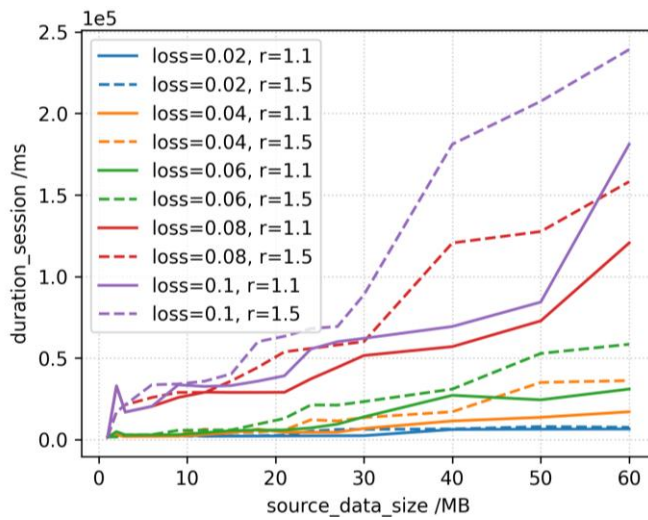


Figure 4. Comparison of session completion time under different packet loss rates and redundancy ratio.

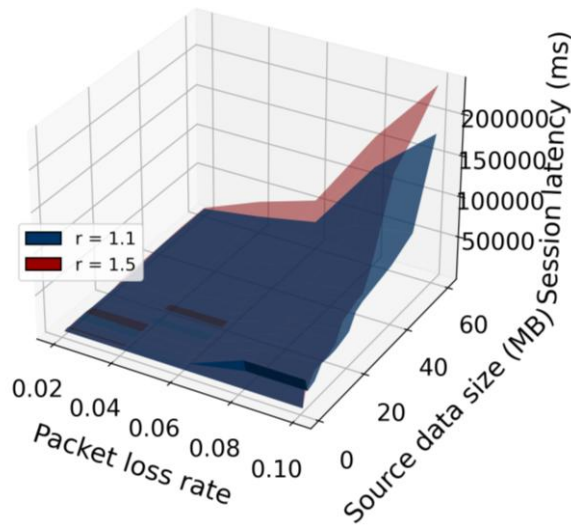


Figure 5. Session completion time under different static redundancy levels

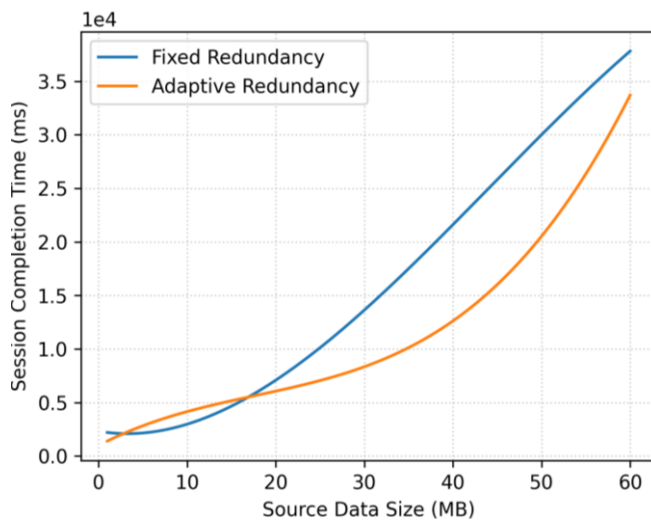


Figure 6. Session completion time: static redundancy vs. adaptive redundancy (ARRCT)

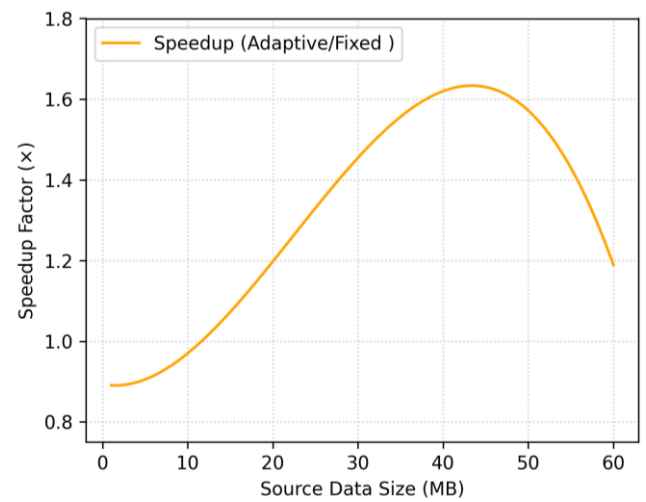


Figure 7. Speedup of ARRCT over Static Redundancy in Transmission Performance

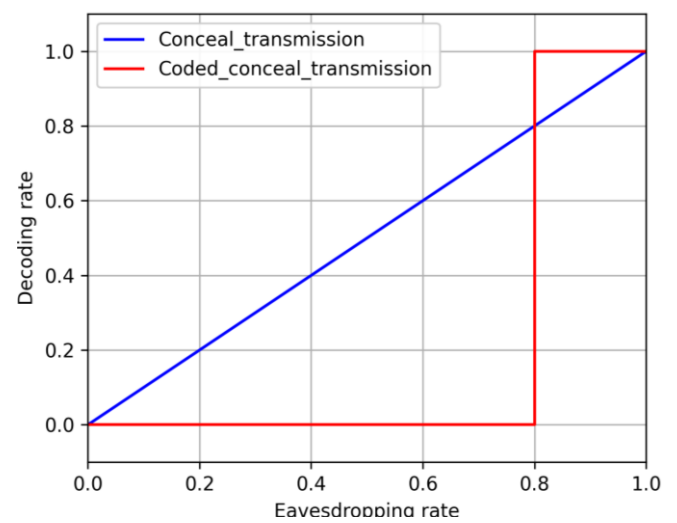


Figure 8. Anti-Eavesdropping Performance: Uncoded vs. Coded Random Routing

V. CONCLUSIONS

The proposed ARRCT mechanism addresses the transmission performance issue—namely, the high latency induced by long-tail effects—in random routing-based covert communication systems, without compromising their existing security properties. With a redundancy adjustment law driven by decoding feedback, ARRCT effectively mitigates the long-tail delay induced by path randomness, while maintaining path diversity and covertness. Meanwhile, Raptor coding increases the difficulty of adversarial decoding and, in combination with random routing that disperses traffic, enhances resistance to both eavesdropping and traffic-analysis attacks, without compromising the original security goals of KRRD. Future work will focus on a more rigorous theoretical study of the ARRCT redundancy control law using stochastic-approximation and control-theoretic tools, characterizing its convergence and its deviation from the static optimal redundancy. In addition, to further strengthen deployability, we

will quantitatively profile the feedback and state-evaluation overhead that becomes non-negligible for small data blocks, including bandwidth consumption, processing time, and their net impact on end-to-end session completion time. We will also report concrete implementation-level metrics for the encoding/decoding cost of Raptor coding in ARRCT (e.g., per-block CPU cycles and processing latency) and integrate such computational delays into the overall completion-time model. Finally, we plan to build a prototype and conduct evaluation under richer attack models and real-world network conditions, enabling a more comprehensive assessment of the performance–concealment trade-off in practical covert random-routing deployments.

ACKNOWLEDGMENT

This work was supported by the self-deployed project of the Institute of Acoustics, Chinese Academy of Sciences Environments (Project No. E555180401).

REFERENCES

- [1] Y. F. Li et al., “Survey on key issues in networks covert channel,” *Ruan Jian Xue Bao / Journal of Software*, vol. 30, no. 8, pp. 2470–2490, 2019 (in Chinese). [Online]. Available: <http://www.jos.org.cn/1000-9825/5859.htm>
- [2] P. Du, J. You, Y. Li, and J. Chen, “An effective moving target defense scheme based on overlay topology mutation and randomized routing dispersion,” *Computer Networks*, 2025, Art. no. 111869. doi: 10.1016/j.comnet.2025.111869
- [3] M. K. Reiter and A. D. Rubin, “Crowds: Anonymity for web transactions,” *ACM Trans. Inf. Syst. Secur.*, vol. 1, no. 1, pp. 66–92, 1998.
- [4] J. Van Den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, “Vuvuzela: Scalable private messaging resistant to traffic analysis,” in *Proc. 25th ACM Symp. Operating Syst. Principles (SOSP)*, 2015, pp. 137–152.
- [5] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, “Riposte: An anonymous messaging system handling millions of users,” in *Proc. IEEE Symp. Security Privacy (SP)*, 2015, pp. 321–338.
- [6] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” in *Proc. 13th USENIX Security Symp. (USENIX Security)*, 2004.
- [7] P. Piotrowska, D. Barron, A. Kate et al., “The Loopix anonymity system,” in *Proc. USENIX Security Symp. (USENIX Security)*, 2017.
- [8] K. Wang, J. You, Y. Li, and J. Chen, “Draughts: A decentralized jump-based system for interactive anonymous communication,” *Electronics*, vol. 14, 2025, Art. no. 4439. doi: 10.3390/electronics14224439.
- [9] F. Prasser et al., “Decentralized reliability estimation for low-latency mix networks,” in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2024.
- [10] A. Shokrollahi, “Raptor codes,” *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.
- [11] T. Hua, Y. Jiang, and K. Xu, “Random routing method in overlay networks,” *Mini-micro Computer Systems*, vol. 34, no. 5, pp. 951–958, 2013 (in Chinese).
- [12] M. Watson, T. Stockhammer, and M. Luby, “Raptor forward error correction (FEC) schemes for FECFRAME,” *IETF, RFC 6681*, Aug. 2012.
- [13] M. Luby, “LT codes,” in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Vancouver, BC, Canada, 2002, pp. 271–280.
- [14] A. Shokrollahi, “Raptor codes,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2004.
- [15] A. Shokrollahi, “Raptor codes,” *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.