Mobile Netw Appl
DOI 10.1007/s11036-011-0333-3

Adaptive Information Coding for Secure and Reliable Wireless Telesurgery Communications

Mehmet Engin Tozal · Yongge Wang · Ehab Al-Shaer · Kamil Sarac · Bhavani Thuraisingham · Bei-Tseng Chu

Abstract Telesurgical Robot Systems (TRSs) have been the focus of research in academic, military, and commercial domains for many years. Contemporary TRSs address mission critical operations emerging in extreme fields such as battlefields, underwater, and disaster territories. The lack of wirelined communication infrastructure in such fields makes the use of wireless technologies including satellite and ad-hoc networks inevitable. TRSs over wireless environments pose unique challenges such as preserving a certain reliability threshold, adhering some maximum tolerable delay, and providing various security measures depending on the nature of the operation and communication environment. In this study, we present a novel approach that uses information coding to integrate both lightweight privacy and adaptive reliability in a single protocol called Secure and Statistically Reliable UDP (SSR-UDP). We prove that the offered security is equivalent to the existing AES-based long key crypto systems, yet, with significantly less computational overhead. Additionally, we demonstrate that the proposed scheme can meet high reliability and delay requirements of TRS applications in highly lossy environments while optimizing the bandwidth use. Our proposed SSR-UDP protocol can also be utilized in similar cyber-physical wireless application domains.

Keywords wireless telesurgery · teleoperation · privacy · statistical UDP · statistically reliable UDP

M. Engin Tozal (✉) · K. Sarac · B. Thuraisingham
Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA
e-mail: engintozal@student.utdallas.edu

K. Sarac
e-mail: ksarac@utdallas.edu

B. Thuraisingham
e-mail: bhavani.thuraisingham@utdallas.edu

Y. Wang · E. Al-Shaer · B.-T. Chu
Department of Software and Information Systems, University of North Carolina Charlotte, Charlotte, NC 28223, USA

Y. Wang
e-mail: yongge.wang@uncc.edu

E. Al-Shaer
e-mail: ealshaer@uncc.edu

B.-T. Chu
e-mail: billchu@uncc.edu

1 Introduction

Telesurgical Robot Systems (TRSs) have been recently the focus of research in academic, military, and commercial domains [37]. Such systems are designed to allow a master (called surgeon controller) to operate on a slave (called surgical robot) located at a distant geographical location. The first generation surgical robots are constructed to perform minimally invasive surgeries on a patient, using a console placed in the operating room. On the other hand, contemporary TRSs address mission critical operations emerging in adversarial environments such as battlefields, underwater, and disaster territories at remote regions [16, 20, 29]. The lack of wirelined communication infrastructure in such fields makes the use of wireless technologies including satellite and ad-hoc networks inevitable. Although these wireless platforms demonstrate varying characteristics [41], design and implementation of a secure and reliable wireless communication conforming to the application tolerable delay and packet loss for TRSs is crucial.

Published online: 13 July 2011

TRSs are categorized as real-time interactive network applications. Similar to the other real-time applications, TRSs are constrained with maximum delay and loss requirements. Recently, a new communications protocol (ITP), has been developed to allow interoperability of surgical robots and controllers [25]. To achieve lowest possible latency, ITP employs light-weight UDP protocol. To deal with small amount of packet losses, ITP relies on repositioning based on the expected continuity of the motion [25]. ITP assumes an acceptable network performance and infrastructure in order to meet reliability, privacy, and security requirements of TRSs. However, these assumptions do not hold when considering insecure, high loss, and high delay nature of wireless environments such as battlefield ad hoc network environments.

To cope with security issues, researchers have extended the ITP to provide basic encryption and authentication using DTLS [34]. However, this approach introduces fixed extra overhead that contributes further to the packet delay problem without addressing the reliability issues. Unlike wirelined networks, the primary source of packet loss in wireless is bit errors instead of congestion. Depending on the type of the operation being performed a TRS session needs to achieve different reliability levels. To illustrate, a mediocre operation might tolerate up to 10% average packet loss (at least 90% average reliability) in an environment providing only 80% reliability on the average, whereas, an intricate surgery might tolerate at most 2% average packet loss in the same environment. As the packet retransmission delay is usually unaffordable in wireless TRSs, using Forward Error Correction (FEC) is mostly the only viable solution. However, just combining traditional crypto techniques such as AES with FEC [35] obtains unacceptable delay for wireless TRSs as the AES encryption delay overhead increases significantly with the traffic redundancy (see Section 7.1). In addition, many of the existing FEC methods based on digital fountain [11, 28, 33, 39] does not provide privacy protection of the transmitted messages and the FEC methods based on network coding [2, 23, 26] are tailored for multicast communication rather than unicast. It should be noted that Luby [28] mentions that LT code (which is a fountain code) packets are generated by some random linear combination of the message packets with the random degree d, where the randomness is shared by the transmitter and the receiver. To some readers, this may be considered as the privacy protection since the receiver could only decrypt the encoded packets with the knowledge of the randomness information. However, the random space could exhaustively be searched within 2^k steps, where k is the number of grouped packets. Thus these randomized fountain LT codes does not provide privacy.

In this paper, we present a novel adaptive information coding scheme to support both confidentiality and adaptive reliability simultaneously. The protocol introduced in this study, *Secure and Statistically Reliable UDP (SSR-UDP)*, demonstrates the fact that security and reliability could be well-integrated into a single protocol in order to accommodate the TRSs requirements in wireless environments. In a typical TRS application, the controller constantly generates messages to be transmitted through a wireless channel and the robot collects and processes the messages at the application layer. SSR-UDP is a light-weight layer located between transport and application layers of the Internet Protocol Suite and it is responsible for addressing security and reliability requirements of TRSs. At the sender side SSR-UDP accumulates k messages while adhering to the application delay constraints and encodes these messages into a batch of n packets ($k < n$). At the receiver side SSR-UDP immediately recovers all k messages given that at least k out of n packets in the batch have been successfully received. SSR-UDP introduces redundancy while translating k messages into n packets in order to maintain required α reliability in the long run over a wireless channel that provides average β reliability such that $0 < \beta < \alpha < 1$. Additionally, our information coding scheme used in SSR-UDP automatically provides both confidentiality and reliability with significantly less delay overhead compared to other standalone crypto approaches such as DTLS [34], TLS/SSL [15], and IPSec [24]. Moreover, our information coding scheme is adjustable to provide varying security key lengths depending on the TRS application requirements.

Our analytical proofs show that suggested privacy scheme can be as strong as AES with 128 and 256 key length, but with significantly less delay overhead (up to 70% for AES-128 and $n = 4, k = 3$). In the reliability side, we have shown that even with high variation of packet loss, the protocol can achieve $\alpha = 99\%$ target reliability threshold over a channel providing only 90% average reliability with a reasonable redundancy ranging between 30–45%. Additionally, we show that our protocol can achieve $\alpha = 98\%$ long run reliability requirement with 86% redundancy over a channel providing 76% reliability on the average.

Although the presented technique can be potentially used with general real-time wireless applications that require both security and reliability, this technique addresses particularly wireless telesurgery because of its unique characteristics which require dynamic adjustment of both security and reliability

simultaneously based on not only network condition but also the type and context of the tele-operation. Therefore, unlike many other real-time wireless applications, telesurgery requires constant tuning of network traffic coding parameters to enable both tele-operation-aware and network-aware adaptation for secure and reliable telesurgery.

The rest of the paper is organized as follows: in Section 2 we present the related work; the proposed information coding technique, cryptographic functions and their analysis is introduced in Sections 3, 4, and 5, respectively; in Section 6 we present dynamic adjustment of message encoding based on the observed loss and delay in the channel; we discuss the experimental results in Section 7; finally, we present conclusions and future works in Section 8.

2 Related work

Since the first telesurgical robot system [18] which was a couple of mechanical hands cabled to a remote handle, many successful works have been done in the field [6, 19, 21]. Brady and Tarn [10], developed a framework for extending teleoperation systems to Internet. Project RAVEN [30] is an implementation of TRSs and it proved that remote surgeries can successfully be realized over the Internet via UDP.

Lum et al. [29], showed that RAVEN can be utilized over transatlantic Internet and wireless radio links. Brett et al., demonstrated an experimental surgical robot in extreme conditions where the installation of wireless networks is not feasible by using unmanned airborne vehicles as a network topology. Finally, another study shows that RAVEN can be deployed in undersea environments [16].

In this study we develop an information coding scheme along with an application layer protocol for addressing both security and reliability in TRSs over wireless links. Our coding scheme is based on adaptive forward error correction [35] (FEC). Many FEC techniques based on digital fountain have been designed in the past [11, 28, 33, 39] for efficient and reliable multicast. In digital fountain technique, the source host divides a given message into k packets and generates a potentially infinite supply of encoding packets from the original k packets. The receiver host reconstructs the original message from any of the received k encoding packets. However, digital fountain techniques are not applicable to TRSs because in these techniques privacy protection is not addressed. Network coding is also used for multicast communication [22, 23, 26, 27]. In a network coding based communication, each intermediate node receives data packets from its incoming edges, combines them by some encoding algorithms, and transmits the encoded data via its outgoing edges. When the receiver receives sufficiently many packets, it could recover the original message with high probability. Although network coding provides a probabilistic framework for increasing network capacity and reliability, it is not applicable to TRSs because (1) it requires the deployment of network coding capability into intermediate nodes, (2) it is suitable for multicast communication rather than unicast, and (3) privacy protection is not addressed.

3 Information coding

In the traditional one-time-pad encryption scheme (Vernam Cipher), one may encrypt a message m with the key a by letting the cipher text $c = am$. This scheme could be extended to achieve reliability and privacy at the same time. Specifically, our information coding scheme is based on a secure key generation function $\mathcal{G}$, which takes a secret key key (shared between the parties) and an arbitrary length string x to return a fixed length (e.g., 256 bits) string $\mathcal{G}(key, x)$. The adversary sees a sequence (x_1, a_1), (x_2, a_2), ..., (x_q, a_q) of pairs of inputs and their corresponding outputs $a_i = \mathcal{G}(key, x_i)$. The adversary breaks the key generation function $\mathcal{G}$ if she can find an input x, not included among $x_1, \ldots, x_q$, together with its corresponding valid output $a = \mathcal{G}(key, x)$. In this paper, our security model is based on the *chosen message attacks*. That is, the adversary is allowed to choose the sequence of inputs $x_1, x_2, \ldots, x_q$. Note that, the other commonly used adversary model is the *known message attacks* where the adversary is not allowed to choose the sequence of messages. The model of chosen message attack is stronger than the known message attack model. In particular, we will use the following model which is adapted from the security model for message authentication code (MAC) [8].

Definition 1 A key generation function $\mathcal{G}$ is (ε, t, q, L)-secure if any adversary that is not given the key key, is limited to spend total time t, and sees the values of $\mathcal{G}(key, \cdot)$ on q inputs $x_1, x_2, \ldots, x_q$ of its choice, each of length at most L, cannot generate the output for an input x $(\neq x_1, \ldots, x_q)$ with a probability better than ε.

In the following, we will present our information coding scheme with a secure key generation function $\mathcal{G}$ as defined in Definition 1. In the next sections, the

construction of $\mathcal{G}$ based on secure hash functions $\mathcal{H}$ (e.g., SHA-256) will be presented and exact analysis techniques [7, 8] will be used to relate the hardness of breaking our information coding scheme to the cryptographic strength of the underlying hash functions.

Let *key* be the ephemeral session key establishment for the secure communication between the sender and the receiver. First, a secure key generation function $\mathcal{G}$ is used to generate the message authentication key $b = \mathcal{G}(key, \text{“HMACKEY”})$.

We may assume that the basic data blocks are elements from a finite field F_p. Let (k, n) be an appropriately chosen pair of integer parameters, which could be configured for specific applications based on the network capacity and reliability requirements as described in Section 6.

Assume that we have a message flow $m_1, m_2, m_3, \cdots$ for delivery. We group these messages into blocks of k and each group will be assigned a sequence number *seq* and will be delivered as one group. In other words, the messages $m_1, \cdots, m_k$ will be put into one group and the messages $m_{k+1}, \cdots, m_{2k}$ will be put into another group.

For each group with the sequence number *seq*, the coefficient matrix is generated by letting $a_{i,j} = \mathcal{G}(key, i||j||seq)$ where $1 \le i \le n$, $1 \le j \le k$, and $||$ is the concatenation operation. Note that this coefficient matrix is only valid for this group of messages with the sequence number *seq*.

Assume that we have one group of messages $(m_1, \ldots, m_k) \in F_p^k$ for delivery. Let

$$\begin{aligned} y_1 &= a_{1,1}m_1 + \ldots + a_{1,k}m_k \\ &\cdots \\ y_n &= a_{n,1}m_1 + \ldots + a_{n,k}m_k \end{aligned} \tag{1}$$

The encoded vector for this group of messages is $(y_1, \ldots, y_n)$ $(n \ge k)$. Using the message authentication key b, the sender will generate the message authentication tags (e.g., using the HMAC scheme [7]) for each of these n messages, and deliver these n encoded messages together with their authentication tags to the receiver.[1]

The receiver will be able to recover the original message vector $(m_1, \ldots, m_k)$ as long as it can receive at least k un-corrupted packets (any k packets will be OK and the order is not important). For example, if the receiver node collects k packets $y_{i_1}, \ldots, y_{i_k}$, then with high probability she could recover the original message as

$$\begin{pmatrix} m_1 \\ m_2 \\ \cdots \\ m_k \end{pmatrix} = \begin{pmatrix} a_{i_1,1} & \cdots & a_{i_1,k} \\ \cdots & \cdots & \cdots \\ a_{i_k,1} & \cdots & a_{i_k,k} \end{pmatrix}^{-1} \begin{pmatrix} y_{i_1} \\ y_{i_2} \\ \cdots \\ y_{i_k} \end{pmatrix} \tag{2}$$

As an example, let $n = 4$ and $k = 3$. We can use four UDP packets to deliver the encoded information and we can tolerate one packet loss. In other words, if the UDP packet loss is at most 25%, we have achieved reliable communication channels using the UDP protocol.

In the following, we use a simple explanation with $k = 3$ to illustrate the advantages of the information coding method.

- If there is only one path, and the information coding method is not exploited, then we need three time unit to deliver the three UDP packets without reliability.
- Let $n = 4$ and assume that there is only one path. Then we can use four UDP packets to deliver the encoded information and we can tolerate one packet loss. At the same time, we achieve perfect message privacy without using any encryption method to avoid the overhead. In other words, if the UDP packet loss is at most 25%, we have achieved reliable communication channels using the UDP protocol.
- Let $n = 4$ and assume that there are two disjoint routing paths. Then our above analysis shows that we can achieve reliable and private packet delivery within two time units with UDP packet if the packet loss is at most 25%.
- In the experiment of the ITP protocol, the command channel UDP packets are delivered at the frequency of 1000 Hz. Now assume that for the underlying network, the UDP packet loss is at most 10%. Then our above analysis shows that with a delivery frequency of 1100 Hz, we can achieve perfect private and reliable communication channels based on UDP packets.

3.1 Relations to network coding

In a random linear network coding scheme, the intermediate nodes choose random coefficients to mix the incoming messages. Network coding scheme is optimized for multicast communications. In our information coding scheme, the source node uses key generated coefficients to mix the messages. From network coding point of view, our scheme could be considered as a

[1] The order of the encoded messages are not important.

private point-to-point version of network coding. From information coding viewpoint, network coding is kind of hop to hop information coding.

The major cost for the information coding and decoding comes from the encoding equation (1) and decoding equation (2). In particular, if the underlying field F_p is larger (e.g., $|p| = 160$), then the information coding and decoding will be very expensive for real time communications. In practice, we can use small integers (e.g., 8-bit integers) coefficients for linear combinations on the field F_p. For example, we may choose a_i from $\{0, \cdots, 255\}$ and each message is 160 bits (that is, an element from the field $F_{2^{160}}$. In particular, the decoding coefficients matrix in the Eq. 2 could be efficiently calculated with table look-ups. Our experiments show that with this technique, the decoding and encoding costs are negligible in real-time communications. It should be noted that similar techniques have been developed for network coding in [17].

When we use information coding over small integers of 8 bits, (n, k) should be chosen in such a way that $8nk$ is large enough (e.g., 128) to avoid exhaustive search attacks.

4 Secure key generation functions

Cryptographic hash functions (e.g., SHA-256) usually do not use any cryptographic keys. Extensive research has been done to design Message Authentication Code (MAC) with keyed cryptographic hash functions. For example, in [7], NMAC and HMAC constructions were designed to use the underlying hash functions as a "black-box" in a way that the hardness of forging an authenticated message can be related to the cryptographic strength of the underlying hash function.

For the convenience of exact analysis, we will use the HMAC-style constructions [7] for our secure key generation functions. Most existing cryptographic hash functions (e.g., SHA-256 and MD5) are based on the iterated construction. The iterated construction is based on a basic component called *compression function* which processes short fixed-length inputs, and is then iterated to hash arbitrarily long inputs. A compression function f accepts two inputs: an internal state variable of length l-bits and a block of data of length b-bits. For SHA-256, we have $l = 256$ and $b = 512$.

An iterated hash function is defined as follows. First an l-bit IV is fixed. Let $x = x_1 x_2 x_3 \cdots x_n$ be the input where x_i's are blocks of length b each and $x_{n+1} = |x|$ be the message length. The output of the iterated hash function $\mathcal{H}$ on x is h_{n+1} where $h_0 = IV$ and $h_i = f(h_{i-1}, x_i)$ for $i = 1, 2, \cdots, n+1$.

Based on the iterated hash functions, the authors in [7] designed hash based message authentication function $\text{HMAC}(key, x)$ as the leftmost t bits of $\mathcal{H}(\overline{key} \oplus \text{opad}, \mathcal{H}(\overline{key} \oplus \text{ipad}, x))$, where opad and ipad are two fixed b-bits constants, and t is pre-determined parameter (e.g., t often takes the value of 64 when HMAC is used with MD5). Our secure key generation function is essentially the HMAC function. We call it key generation function and use a different notation $\mathcal{G}$ since our emphasis here is on key generation instead of message authentication and we do not need to truncate the outputs of the external-layer hash function. In particular, let $\mathcal{H}$ be a hash function which takes arbitrary length inputs and outputs l-bits strings. Then the secure key generation function is defined as:

$$\mathcal{G}(key, x) = \mathcal{H}(\overline{key} \oplus \text{opad}, \mathcal{H}(\overline{key} \oplus \text{ipad}, x))$$

where ipad is "the byte 0x36 repeated $b/8$ times" and opad is "the byte 0x5C repeated $b/8$ times" which is similar to the HMAC standard.

The authors in [7] show that their HMAC construction is secure if the underlying hash function is collision resistant and if the keyed compression f is a secure MAC on messages of b bits, where the keyed compression function f_k is defined as $f_k(x) = f(k, x)$ for $|k| = l$ and $|x| = b$. For the security model in Definition 1, we could get the following theorem.

Theorem 1 (Adapted from Bellare et al. [7]) *If the keyed compression function f is an (ε, q, t, b)-secure MAC on messages of b bits, and $\mathcal{H}$ is collision-resistant, then $\mathcal{G}$ is an (ε, t, q, L)-secure key generation function.*

Proof This follows from [7, Theorem 4.1] and the discussion in [7, Section 5.2]. □

5 Security analysis of information coding scheme

In order to relate the hardness of our information coding scheme to the cryptographic strength of the underlying hash function, we first show that if the underlying hash function $\mathcal{H}$ is collision resistant, then it is hard for the attacker to generate a linear combination of outputs of the key generation function $\mathcal{G}$ on two inputs. Notice that this condition is different from output generation for the key generation function since the attacker may not need to generate the outputs for two inputs in order to generate a linear combination of them somehow.

Lemma 1 *If $\mathcal{G}$ is a (ε, t, q, L)-secure key generation function, then for any adversary that is not given the key key, is limited to spend total time $t - 1$, and sees*

the values of the function $\mathcal{G}(key, \cdot)$ computed on $q-1$ inputs $x_1, x_2, \ldots, x_q, x_{q-1}$ of its choice, each of length at most L, cannot find a tuple $(\gamma_1, \gamma_2, \gamma_3, x, x')$ for which $|\gamma_1| + |\gamma_2| \neq 0$ and $\gamma_1 \mathcal{G}(key, x) + \gamma_2 \mathcal{G}(key, x') = \gamma_3$ with probability better than ε. Notice that here, the adversary does not need to find the values of $\mathcal{G}(key, x)$ or $\mathcal{G}(key, x')$.

Proof Let us fix the parameters q, t, L first. For a contradiction, we assume that there is an attacker A_γ with success probability ε to generate a linear equation $\gamma_1 \mathcal{G}(key, x) + \gamma_2 \mathcal{G}(key, x') = \gamma_3$ after $q-1$ queries on $\mathcal{G}(key, \cdot)$ at its own choice of the inputs. In the following, we will design an adversary $A_\mathcal{G}$ that forges the output of $\mathcal{G}(key, \cdot)$ on an input by spending q queries and time t with a success probability of ε. This shows that any adversary that tries to generate the above mentioned linear equation with the above resources has a probability of success of at most ε. This proves the theorem.

We construct $A_\mathcal{G}$ by oracle access to A_γ. Note that A_γ works as follows. It queries the key generation function $\mathcal{G}$ adaptively on inputs $x_1, \ldots, x_{q-1}$ and gets the responses $\mathcal{G}(key, x_1)$, ..., $\mathcal{G}(key, x_{q-1})$. Note that the choice of x_i by A_γ may depend on the choices of $x_1, \ldots, x_{i-1}$ and the responses $\mathcal{G}(key, x_1)$, ..., $\mathcal{G}(key, x_{q-1})$. It finally outputs $(\gamma_1, \gamma_2, \gamma_3, x, x')$. If $x \neq x_i, x' \neq x_i$ for $i = 1, 2, \ldots, q-1$, $|\gamma_1| + |\gamma_2| \neq 0$, and $\gamma_1 F_k(m) + \gamma_2 F_k(m') = \gamma_3$ then the attack succeeds, otherwise it fails.

The goal of $A_\mathcal{G}$ is to forge $\mathcal{G}(key, \cdot)$, by querying $\mathcal{G}(key, \cdot)$ on inputs that $A_\mathcal{G}$ chooses. We need to describe how $A_\mathcal{G}$ chooses the messages, how it gets responses, and how it outputs a message x and an authentication tag on it.

$A_\mathcal{G}$ first activates A_γ which produces queries to the function $\mathcal{G}(key, \cdot)$ that will be answered by $A_\mathcal{G}$ via oracle queries to $\mathcal{G}(key, \cdot)$ again. In particular, A_γ adaptively chooses messages $x_1, \ldots, x_{q-1}$. For each of the messages x_i, $A_\mathcal{G}$ queries $\mathcal{G}(key, \cdot)$ for the answer $\mathcal{G}(key, m_i)$. A_γ finally outputs $(\gamma_1, \gamma_2, \gamma_3, x, x')$. We distinguish the following cases:

1. A_γ fails the attack. That is, $x = x_i$ or $x' = x_i$ for some $i = 1, 2, \ldots, q-1$ or $\gamma_1 = \gamma_2 = 0$. In this case, $A_\mathcal{G}$ fails to output the forgery.
2. $x \neq x_i$ and $x' \neq x_i$ for all $i = 1, 2, \ldots, q-1$ and $\gamma_1 = 0$ or $\gamma_2 = 0$. Without loss of generality, we assume that $\gamma_1 = 0$. In this case, $A_\mathcal{G}$ outputs $(x, \gamma_3/\gamma_1)$ as the forgery.
3. $x \neq x_i, x' \neq x_i$ for all $i = 1, 2, \ldots, q-1$ and $\gamma_1 \gamma_2 \neq 0$. In this case, $A_\mathcal{G}$ queries $\mathcal{G}(key, \cdot)$ for the answer $\mathcal{G}(key, x)$ and outputs $\left(x', \frac{\gamma_3 - \gamma_1 \mathcal{G}(key, x)}{\gamma_2}\right)$ as the forgery.

Now we analyze the success probability ε of the above algorithm $A_\mathcal{G}$. $A_\mathcal{G}$ fails whenever A_γ fails. That is, A_γ fails to output a successful tuple $(\gamma_1, \gamma_2, \gamma_3, x, x')$ for which $\gamma_1 \mathcal{G}(key, x) + \gamma_2 \mathcal{G}(key, x') = \gamma_3$. Thus the failure probability of $A_\mathcal{G}$ is bounded by the failure probability ε of A_γ. □

Now we are ready to show the exact security of the information coding scheme.

Theorem 2 *If $\mathcal{G}$ is a (ε, t, q, L)-secure key generation, function, then for any adversary that is not given the key key, is limited to spend total time $t - nk$, and sees the values of the function $\mathcal{G}(key, \cdot)$ computed on $q - nk$ inputs $x_1, x_2, \ldots, x_q, x_{q-nk}$ of its choice, each of length at most L, learns zero information of the messages $(m_1, \ldots, m_k)$ in the information coding scheme.*

Proof It should be noted that in the information coding scheme, there are $(n+1) \times k$ unknowns and only n packets are delivered along the paths. These n packets correspond to n equations. By Theorem 1, the adversary may not be able to generate further $(n+1) \times k - n$ equations by oracle access to the secure key generation function. Thus without the knowledge of the ephemeral session key *key*, the adversary learns zero information of the original message vector $(m_1, \ldots, m_k)$. □

Corollary 1 *If the keyed compression function f is an (ε, q, t, b)-secure MAC on messages of b bits, and $\mathcal{H}$ is collision-resistant, then the adversary learns zero information of the messages $(m_1, \ldots, m_k)$ in the information coding scheme.*

Proof This follows from Theorems 1 and 2. □

6 Dynamic adjustment of (n, k)

The performance of the proposed information encoding scheme depends on the careful selection of k and n values so as to bound application experienced loss ratio as well as to respect the security requirements of the encoding scheme. In this section, we discuss the factors that affect the selection and dynamic adaptation of k and n for optimal information coding.

In telesurgery, the controller constantly generates a single message per unit time, k of these messages are accumulated and wrapped into n code packets via information encoding ($k < n$), and all n code packets

are streamed out over the next k unit time period. The process of k message accumulation; n packet encoding; and their dispatch is called a "batch transmission". A communication session between a sender and a receiver consists of numerous batch transmissions depending on k. That is, for a session with m messages in total, the number of batches would be $\lceil {}^{m}/_{k} \rceil$ for a fixed k.

The motivation behind translating k messages into n code packets ($k < n$) is to ensure with probability α that at least k of the n code packets are successfully transmitted through a lossy logical channel. The α probability could be a constant or a varying parameter depending on the loss tolerance of the application for the next batch. The receiver can perfectly recover the original k messages as long as it receives at least k of the n code packets.

In the following, we analyze constraints imposed on n and k and discuss how to balance their values regarding the constraints.

6.1 Constraints analysis of n

In this section, we discuss how to choose a proper n value under the requirements dictated by the application and constraints imposed by the networking infrastructure. In order to convey a healthy discussion regarding the effect of the varying n values on the application and on the networking resources, we need to abstract the communication between the sender and the receiver. Remember that, loss in wirelined networks is mostly due to congestion, however, in wireless environments loss is mostly due to bit errors. In our model we have a sender and a receiver communicating through a logical channel with packet loss probability q and packet losses are statistically independent [5, 12]. A packet is successfully transmitted with probability $p = 1 - q$. Let R be a discrete random variable denoting the number of successfully transmitted packets out of a batch of n code packets. Then, R has a binomial distribution with parameters n and p, i.e., $R \sim Binomial(n, p)$. Transmission of a batch of n code packets is regarded as a "successful batch" as long as at least k code packets make it to the receiver through the lossy channel. Hence, probability that a batch will be successful is calculated as:

$$\begin{aligned} P\{Successful\ Batch\} &= P\{R \geq k\} \\ &= \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned} \tag{3}$$

Given k messages and a logical channel with successful packet transmission probability p, our objective is to find a value n such that the probability that at least k out of n code packets has been received is α, i.e., $P\{Successful\ Batch\} = \alpha$. Let m be the total messages generated by the application and $\langle k \rangle$ be the average k, then the expected value of successfully transmitted SSR-UDP application messages is $\langle k \rangle \frac{m}{\langle k \rangle} \alpha$. For conventional UDP, it is mp. As a result, through a careful selection of k and n, we can achieve α reliability over a channel providing p reliability ($\alpha > p$) in the long run. Equivalent to Eq. 3, we can work with the expression $P\{Failed\ Batch\} = 1 - \alpha$.

$$\begin{aligned} P\{Failed\ Batch\} &= P\{R \leq k-1\} \\ &= \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \\ \Rightarrow \quad 1 - \alpha &= \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \\ &\text{such that } p < \alpha \end{aligned} \tag{4}$$

Solving Eq. 4 for n would give us the proper number of code packets needs to be sent in order to yield a successful batch with α probability over the channel. However, Eq. 4 does not have a closed form solution. To develop a "numerical" solution we resorted to normal approximation[2] to the binomial distribution R along with continuity correction. Let X be a normally distributed random variable with parameters μ and σ^2, i.e., $X \sim Normal(\mu, \sigma^2)$, such that $\mu = np$ and $\sigma^2 = np(1-p)$. Let Z be the standard normal form of X, i.e., $Z = (X - \mu)/\sigma$. Then

$$\begin{aligned} P\{Failed\ Batch\} &= P\{R \leq k-1\} \\ \Rightarrow \quad 1 - \alpha &\approx P\left\{X \leq k - \frac{1}{2}\right\} \\ &= P\left\{Z \leq \frac{k - \frac{1}{2} - \mu}{\sigma}\right\} \\ \Rightarrow \quad z_{1-\alpha} &= \frac{k - \frac{1}{2} - np}{\sqrt{np(1-p)}} \end{aligned} \tag{5}$$

Equation 5 is obtained by replacing μ and σ with their real values np and $\sqrt{np(1-p)}$, respectively. $z_{1-\alpha}$ is the left quantile function of the standard normal

[2] As a rule of thumb, normal approximation to the binomial distribution improves whilst $np \geq 5$ and $n(1-p) \geq 5$. Our empirical results, however, show that applying continuity correction significantly reduces the approximation error to tolerable values for small n and large p values.

distribution Z. Solving Eq. 5 for n results in the final value for n in terms of k, p, and α as follows;

$$n = \frac{2k - 1 + z_{1-\alpha}^2(1-p)}{2p} - \frac{z_{1-\alpha}\sqrt{z_{1-\alpha}^2(1-p)^2 + (1-p)(4k-2)}}{2p}$$
$$\text{such that } 0 < \alpha < 1 \text{ and } k \geq 1 \quad (6)$$

Given that k and α are constants in Eq. 6, as p goes to zero, i.e., packet loss probability ($q = 1 - p$) goes to one, n goes to infinity. Put in other words, as the channel becomes more and more unreliable, the number of generated code packets in order to sustain α success rate increases without any bound. At a heavily loaded channel with small buffering capacity as n increases, p will decrease which in turn cause n to increase and so on. We take care of this potential ill-behavior (spinning effect) in our algorithm by reducing k (which implicitly reduces n) as we experience consecutive decrements in p.

6.2 Constraints analysis of k

In this section, we discuss how to adjust k with respect to the application requirements and resource utilization. Since the number of packets should be an integer, n in Eq. 6 is rounded up and this introduces traffic waste with an expected value of 0.5 packet/batch. For a session with m application messages, if we use $k = 1$, then it takes m batches to complete data transfer resulting in a waste of $m/2$ packets, i.e., 50% unnecessary increase in traffic load into the network.

The above analysis suggests that increasing k reduces waste as it decreases the number of batches during a session. However using very large k values introduces two issues: (1) real-time data have an application-dependent delay threshold and increasing k by accumulating more messages will likely to increase the delay significantly as discussed in Section 7.1, and (2) as k increases, n also increases and sending large batches might potentially fill an intermediate queue and cause more packet drops. In our algorithm we dynamically adjust the value of k based on the real-time delay constraints of the application and observed loss rate of the channel. To encode a group of k packets in our information coding scheme, we need to generate a secret coefficient matrix with kn elements, each of the matrix elements consists of 8 bits. Thus an exhaustive search of this secret coefficient matrix takes time 2^{8kn}. In order to achieve RSA-1024 level of security (which is approximately 80 bits security) we require $8kn \geq 80$ (i.e., $kn \geq 10$). Given that $n \geq k + 1$, this requirement translates into $k(k + 1) \geq 10$ giving us a lower bound on k, as $k \geq 3$. In our algorithm, we explicitly check that $k \geq 3$ and hence ensure that this security requirement is always satisfied.

In summary, n is bounded below by k and optimized with respect to p and α, and k bounded below by 3. Asymptotically, n and p are inversely proportional, i.e., $n \propto p^{-1}$. Moreover, decreasing k decreases communication delay but contributes to waste traffic. On the other hand, increasing k increases communication delay and potentially decreases p.

Algorithm 1 dynamically controls (k, n) values. We assume that there are two channels; namely a data channel from the sender to the receiver and a feedback channel in the reverse direction. The sender receives the success fraction of the packets that are sent in a batch through the feedback channel. Algorithm 1 is executed after each feedback message to calculate the next values of k and n.

Algorithm 1 Dynamic adaptation of k, n

Input: k { current value of k }
Input: α { application reliability requirement }
Input: p_{next} { estimated packet success probability for the next batch }
Input: p_{last} { estimated packet success probability for the previous (last) batch }
Input: Δp_0 { previous amount of change in packet success probability }
Input: d_{next} { estimated channel delay for the next batch }
Input: d_{max} { maximum tolerable real time application delay }
Input: d_{imsg} { delay between generation of two application messages }
Output: k, n { $3 \leq k < n$ }
1 $\Delta p_1 \leftarrow p_{next} - p_{last}$ { current amount of change in packet success probability}
2 **if** $\Delta p_0 < 0$ **and** $\Delta p_1 < \Delta p_0$ **then**
3 decrease k
4 **else**
5 **if** $|\Delta p_1| \leq \mathcal{E}_1$ **then**
6 do not change k
7 **else if** $\Delta p_1 \geq 0$ **then**
8 increase k
9 **else**
10 decrease k
11 **end if**
12 **end if**
13 $d_{enc} \leftarrow$ *estimate encoding delay*
14 $d_{dec} \leftarrow$ *estimate decoding delay*
15 $k_{max} \leftarrow \theta((d_{max} - d_{enc} - d_{next}(1 + \mathcal{E}_2) - d_{dec} + d_{imsg})/d_{imsg})$
16 **if** $k > k_{max}$ **then**
17 $k \leftarrow k_{max}$
18 **end if**
19 $n \leftarrow$ *calculate using Eq. 6*
20 **return** $k, \lceil n \rceil$

The algorithm controls n by changing k in order to preserve a successful batch transmission with α probability. If there are two consecutive negative changes in the amount of packet success probability p, we anticipate growing unreliability in the channel and decrease k as suggested at lines 2 and 3 of Algorithm 1. On the other hand, if the current reliability change is not significant we do not update the value of k; if the reliability has significantly increased we utilize the channel by raising k; and if the reliability has significantly

diminished, we avoid from more potential code packet losses as well as the spinning effect by decreasing k as demonstrated at lines 5–11. k is set to 3 at the beginning of the communication session. In our experiments we incremented k by 20% and decremented it by 40% each time. Compared to additive increase multiplicative decrease model, 20–40% increment-decrement model utilizes the channel more aggressively. Lines 13 and 14 requires estimating the encoding and decoding delays in terms of the physical time, respectively. Note that encoding/decoding delay depends on many factors including implementation, programming language preference, CPU power, and whether the machine has a dedicated crypto hardware. At lines 15–18, we check whether the suggested k violates real time nature of the application and re-adjust its value in case it does. The factor $0 < \theta < 1$ lets us gain some room in terms of time and stream the batch over that time instead of sending it as a burst. θ is set to 0.6 in our simulations. At line 19 we calculate the value of n using Eq. 6.

Finally, the algorithm does not dictate any method for calculating p_{next} (estimated packet success probability) and d_{next} (estimated channel delay) for the next batch. One can use exponential moving average, last observation, or the highest observed value which could be obtained through feedback messages [5, 9, 36]. Nevertheless, the error in these estimations affects optimization of k and n.

Note that, both delay and loss probabilities are based on varying channel properties as well as the estimation that we do for the next batch before dispatching it. Even if in the case of constant channel loss probability model (Section 7.2), setting the packet loss probability to a fixed value does not necessarily mean that each batch experiences the same amount of loss. Some batches may experience a few packet losses while some other batches may not experience any loss at all. Moreover, estimating the loss probability of the next batch is based on historical data which always possesses a random amount of estimation error. Hence, another piece of randomness comes from the estimation. As a result, having a constant value of p_{next} during an entire SSR-UDP session is negligibly small.

7 Performance evaluations

7.1 Theoretical analysis of information coding performance

In this section, we briefly discuss the theoretical performance of our solution against other solutions such as TLS and DTLS. The ephemeral session key establishment procedures for our scheme and DTLS are session key establishment based on public key certificates. Thus they have approximately the same performance. Furthermore, the ephemeral session key is established only once per session and will not have too much impact on the real time performance.

In the DTLS/TLS protocol, the communication content is encrypted via symmetric ciphers such as AES or stream ciphers (for TLS only). In our scheme, both the sender and the receiver need to generate the coefficient matrix via the secure hash function and carry out n linear operations for the sender and k linear operations for the receiver in the field F_p as specified in the Eqs. 1 and 2. Furthermore, the receiver needs to carry out a $k \times k$ matrix inversion over the small integers of 8 bits as specified in the Eq. 2. For any given sequence number seq, both the sender and the receiver can compute the values of $\alpha_{i,j} = \mathcal{G}(key, i||j||seq)$ for $1 \leq i \leq n$ and $1 \leq j \leq k$ in advance. In our scheme, the values of (n, k) could be dynamically adjusted during the protocol execution based on network performance. However, both the sender and the receiver could choose a reasonable large (n_0, k_0) and pre-compute the values of $\alpha_{i,j}$ for all $1 \leq i \leq n_0$ and $1 \leq j \leq k_0$ in advance. Then, in the real-time execution of the protocol with $n \leq n_0$ and $k \leq k_0$, both the sender and the receiver have all the required values of $\alpha_{i,j}$ in hand. With these pre-computations, the real time required operations for the sender is n linear operations and the operations for the receiver is k linear operations and a $k \times k$ matrix inversion.

AES encryption/decryption operations are based on 128-bit blocks. For each AES encryption/decryption operation, we need to carry out one key expansion (scheduling) and several rounds of iterative cipher operations. For the CBC mode, the key expansion (scheduling) is done once for a given key though for other modes such as counter mode, the key expansion (scheduling) needs to be done for each 128-bit block of data. For the reason of convenience, in our following comparison, we assume that DTLS (or TLS) uses AES-128 with CBC mode.

For AES-128, each encryption includes 10 rounds and each round includes four steps: *substitute bytes*, *shift rows*, *mix columns*, and *add round keys*. If we ignore the fast operation *shift rows*, AES-128 needs to carry out 480 linear operations (160 mix-columns, 160 add-round-keys, and 160 substitute bytes which is S-Box look-up) over the finite field F_{2^8} for each encryption (decryption) of a 128-bit block.

AES-128 needs to carry out approximately 480 linear operations over the finite field F_{2^8} for each encryption (decryption) of a 128-bit block. For our information coding scheme, the performance depends on

packet size and the choice of (n, k). For the reason of convenience for comparison, we assume that each packet is 128 bits (in practice, we could always divide big size packets into small packets of 128 bits) and the $\alpha_{i,j}$ belongs to the finite field F_{2^8}. With these assumptions, for k packets of 128-bit size data, the sender needs to carry out $2nk$ linear operations over $F_{2^{128}}$, and the receiver needs to carry out $2k^2$ linear operations over $F_{2^{128}}$ and a $k \times k$ matrix inversion over F_{2^8}. For the $k \times k$ matrix inversion, the trivial Gaussian Elimination method takes k^3 operations, Strassen algorithm takes less than $5k^{2.807}$ operations, and Commpersmith-Winograd algorithm takes $O(k^{2.376})$ operations over F_{2^8}.

There are other fast methods such as Strassen algorithm [40] which takes less than $5k^{2.807}$ operations over F_{2^8} and the fastest Commpersmith-Winograd [13] algorithm which takes $O(k^{2.376})$ operations over F_{2^8}.

For a modular operation αx with $x \in F_{2^{128}}$ and $\alpha \in F_{2^8}$, we can write it as $\alpha(x_{15}2^{15\times 8} + \cdots + x_1 2^8 + x_0)$ where $x_i \in F_{2^8}$. Thus each linear operation over $F_{2^{128}}$ can be approximately counted as 35 operations over F_{2^8}. For fast integer multiplication implementation package such as MIRACL [38], it is feasible to achieve the above approximation. That is, we can carry out one linear operation over $F_{2^{128}}$ at the cost of 35 operations over F_{2^8}.

In Table 1, we list the comparison data for several examples with the above assumption using various values of k (3, 5 and 7) that obtain a reasonable delay bound (less than 150ms one-way), and various values of n that represent up to 50% redundancy. The AES with FEC column presents the numbers of operations over F_{2^8} plus the estimated FEC overhead for each of the sender and the receiver based on Raptor coding which is one the fastest coding technique with a computation complexity of $n * packetsize$ [39], the total of Info-Coding Sender represents the approximate total numbers of linear operations over F_{2^8}, and the total Info-Coding Receiver represents the approximate total numbers of linear operations over F_{2^8}.

Table 1 Performance comparison of DTLS (AES based) and info-coding for k data messages in terms of the number of operations

(n, k)	Redundancy %	AES with FEC	Info-coding sender	Info-coding receiver
(4,3)	25	2432	840	657
(5,3)	40	3040	1050	657
(6,3)	50	3648	1260	657
(6,5)	16	3648	2100	1875
(7,5)	28	4256	2450	1875
(8,5)	37	4864	2800	1875
(8,7)	12	4864	3920	3773
(9,7)	22	5472	4410	3773
(10,7)	30	6080	4900	3773

Table 2 Normalized cost comparison (operations/redundancy) of DTLS (AES based) and info-coding

	Redundancy range	AES with FEC cost	Info-coding cost
$k = 3$	25–40%	41	14
$k = 3$	40–50%	61	21
$k = 5$	16–28%	51	29
$k = 5$	28–37%	68	39
$k = 7$	12–22%	61	49
$k = 7$	22–30%	76	61

Tables 2 and 3 show the significant advantage of our approach over AES over FEC. In particular, Table 3 shows when $k = 3$ and the redundancy increases from 25% to 50%, AES results in 59.8% increase in computational overhead because the number of AES operations increases from 41 to 61 per unit redundancy (1%), as shown in Table 2. However, our coding approach shows that just 20% and 0% overhead increase under the same conditions in the sender and receiver sides respectively. Our analysis also shows that the advantage of our approach decreases as both k and n increase. However, since the buffering overhead must be minimized for bounded delay, it is very unlikely for TRS to use large value of k particularly in wireless networks in order to achieve less than 150 ms end-to-end delay.[3]

For the Crypto++ Library [14], benchmarks for AES encryption/decryption were obtained for a computer with AMD Opteron 8354 2.2 GHz processor under Linux. The library was compiled with GCC 4.1.2 using -O3 optimization, and x86-64/MMX/SSE2 assembly language routines were used for integer arithmetic (see [14] for details). With that configuration, AES encryption/decryption speed is 102 MB per second. If we take the microsecond ($\mu s = 10^{-6}$ s) on the above configured Linux computer as the unit time, then we can carry out 102 Bytes AES encryption (or 3060 F_{2^8} field operations) in one unit time. Based on the analysis in previous paragraphs, for (n, k)-based information coding, the sender could process $\frac{3060\times 16}{35\times 2n} = \frac{699}{n}$ bytes data per unit time and the receiver could process $\frac{3060\times 16}{k^2+70k} = \frac{48960}{k(k+70)}$ bytes data per unit time. For example, with the above Linux configuration, and $n = 4, k = 3$, the sender could process 175MB data per second and the receiver could process 223MB data per second. For

[3]Note that each packet in TRS requires at least $1ms$ to be produced.

Table 3 Comparison of overhead percentage of DTLS (AES based) and info-coding for k data messages

	AES overhead %	Info-coding overhead %
$k = 3$	59.8	20.0
$k = 5$	66.5	37.9
$k = 7$	75.0	60.3

$n = 9, k = 7$, the sender could process 78MB data per second and the receiver could process 91MB data per second. Moreover, Table 4 shows sample benchmarks using the above (n, k) combinations and the corresponding processed data in MB per second.

7.2 Evaluation of secure and statistically reliable UDP

In this section we present our simulation-based evaluations of SSR-UDP. Our simulation setup consists of a sender, a receiver, and a logical channel between them. The logical channel is subject to a random delay and random loss model conforming to a distribution. Because the simulated environment is straightforward and logical channel loss and delay models can be used to hide the complexities of the underlying infrastructure as well as traffic, we implemented a custom-build simulator instead of using a more complicated simulator package such as ns-2. We make our source code and binary executable of SSR-UDP simulator publicly available on our web site [1].

Our evaluation metrics include reliability, redundancy, and waste and our parameters are application demanded reliability (α), channel loss distribution, and channel loss variance. Our experiments include a sender sending real-time traffic to a receiver who sends feedback information to adjust n and k. We used three different channel loss probability and tracking models: (1) Uniform distribution with no loss estimation (UPNE), assuming unpredictable random channel loss, for which we use the maximum observed loss rate as the channel's loss rate, (2) Triangular distribution with regular loss estimating (TPRE) using exponentially moving averages method assuming some persistence in average, (3) Constant channel loss probability with perfect loss estimation (CPPE), to represent the best case scenario. In case of TPRE, we constantly set the mode of the triangular distribution to the last generated random value to simulate wireless environment losses. Although our worst case (UPNE) and base case (CPPE) scenarios are unrealistic, we use them to demonstrate the upper and lower bounds. We assume the delay tolerance bound is 150 msec and RTT is changing between [40–60] msec.

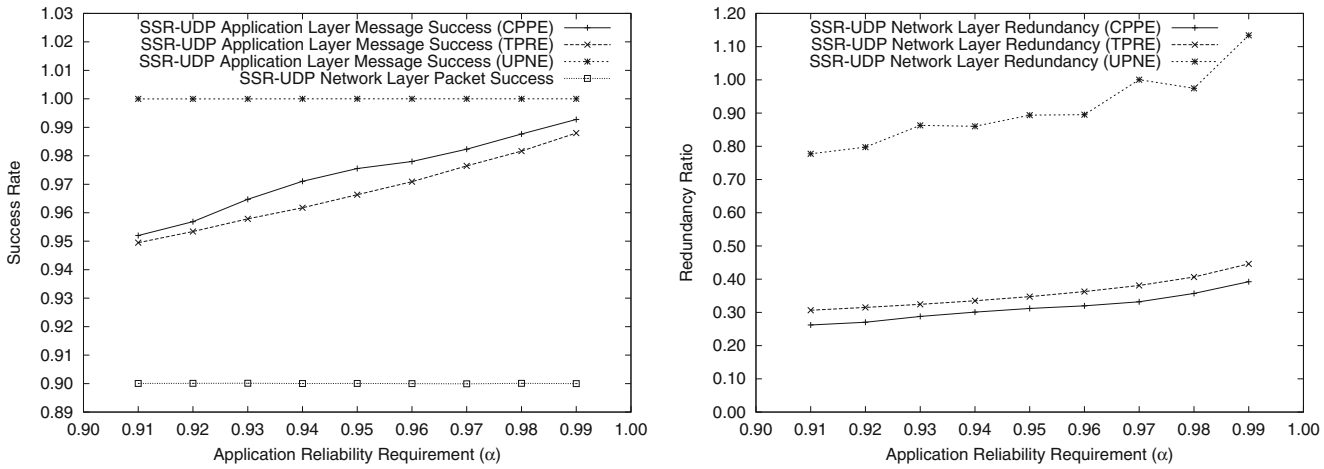
Note that, in all simulations we provide network layer reliability, redundancy, and waste as the base case. The network layer base case metrics refer to the values obtained via conventional User Datagram Protocol (UDP). Network layer metrics not only provide a reference point to SSR-UDP but also serve as a comparison between SSR-UDP and UDP. Hari et al. [4] and Bakre et al. [3] provide methods reducing retransmissions to improve TCP protocol in wireless environments. Reliable UDP (RFC 1151) is an attempt to guarantee packet order delivery and improve quality of service by implementing windowing, acknowledgement, and flow control mechanisms on top of UDP. Other studies suggest FEC based protocols tailored for multimedia communications in wireless environments [31, 32]. However, SSR-UDP is the only protocol addressing both security and reliability simultaneously to the best of our knowledge. Therefore, comparing SSR-UDP directly with any of the mentioned protocols would not be a fair evaluation.

Empirical success rate analysis: In this part we ran a set of simulations demonstrating how well we empirically achieve the required application success rate α and analyze its cost. At each simulation the sender generates 10 million application messages with a rate of 1 message/msec.

Figure 1a and 1b show the empirical message success ratio and its related redundancy with respect to the changing values of α in the interval [0.91–0.99], respectively. In Fig. 1a SSR-UDP achieves at least the required reliability α under all loss models. The top line in Fig. 1a shows the empirical success rate with UPNE. For UPNE, we modeled the channel loss with $Uniform(0.05, 0.15)$ distribution with mean 0.10 and we used the highest loss rate that we have observed at any time as the loss estimation $(1 - p)$ in our algorithm. Since it takes the most dramatic action with respect to

Table 4 Benchmark comparison in MB/s of DTLS (AES based) and InforCoding with Linux on AMD Opteron 8354 2.2 GHz

(n, k)	AES-128 with redundancy	Info-coding sender	Info-coding receiver
(4,3)	76.5	175	223
(5,3)	61.2	140	223
(6,3)	51	116	223
(6,5)	85	116	130
(7,5)	72.8	100	130
(8,5)	63.75	87	130
(8,7)	89.25	87	91
(9,7)	79.33	78	91
(10,7)	71.4	70	91



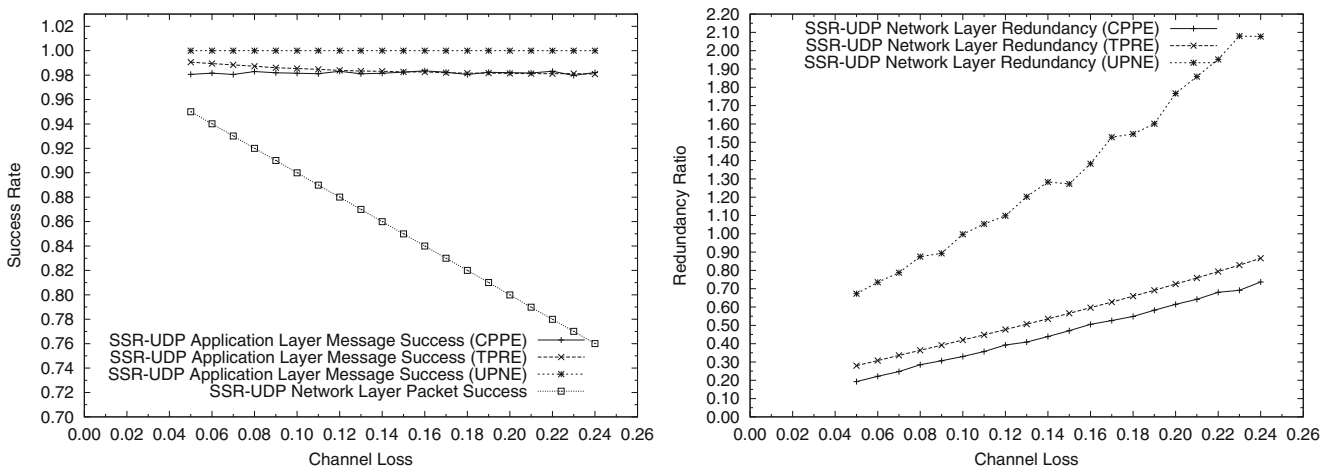
(a) (b)

Fig. 1 Behavior of success rate and redundancy with respect to application reliability requirement (α). **a** Application reliability requirement (α) vs. success ratio. **b** Application reliability requirement (α) vs. redundancy

channel loss, it achieves (very close) to 100% reliability at each application demanded reliability α. On the other hand, Fig. 1b shows that UPNE introduces the most redundancy into the network. The second line presents CPPE. For this simulation the channel loss probability is set to 10% and the algorithm optimized k and n according to this pre-known loss rate. The third line shows TPRE. For this simulation we modeled channel loss probability with triangular distribution over the range [0.05, 0.15] with mean at 0.10. Besides, in order to reduce the batch losses due to channel loss underestimation, we constantly introduced a 0.03 points channel loss overestimation factor. Clearly, the demanded reliability along with the best redundancy is achieved with CPPE. However, TPRE a more realistic model, also attains the required reliability with a redundancy changing between 30% and 44%.

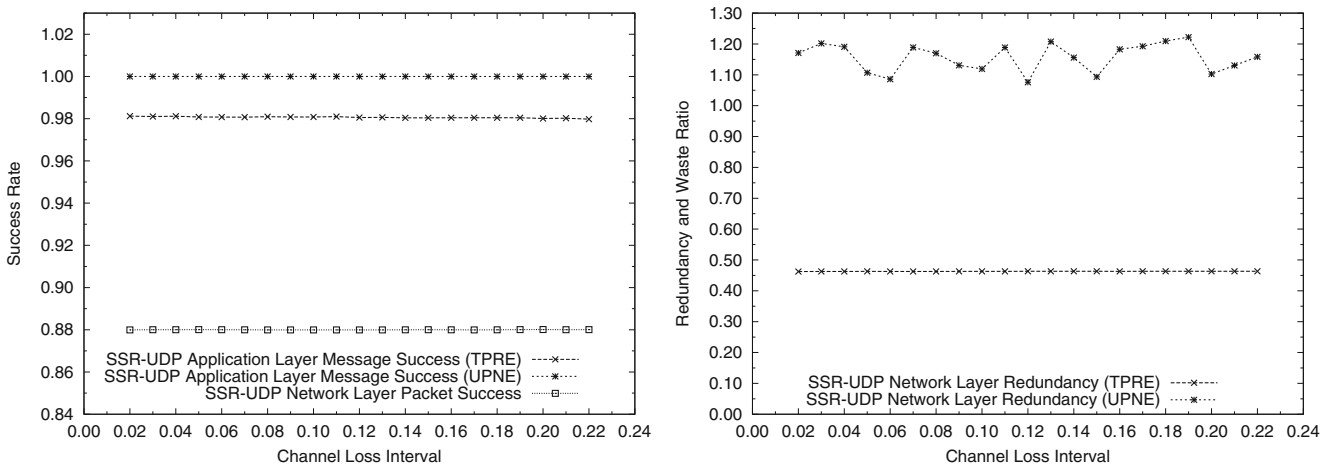
The bottom line in Fig. 1a shows that since all channel models have 10% average loss rate, SSR-UDP experiences 10% packet loss at the network layer. Additionally, we observed the waste (due to upper rounding of n) to be around 4% for all models which confirms that waste is a function of the number of batches and our simulations have generated around 8% of the total messages as batches.

Finally, we ran a set of simulations to see how redundancy changes with a relatively high level reliability requirement [42]. We achieved 99.9% success rate with 79% redundancy for the above TPRE loss model. We also observed that as the reliability requirement



(a) (b)

Fig. 2 Behavior of success rate and redundancy with respect to channel loss ($1 - p$). **a** Channel loss ($1 - p$) vs. success ratio. **b** Channel loss ($1 - p$) vs. redundancy



(a) (b)

Fig. 3 Behavior of success rate, redundancy, and waste with respect to channel loss interval. **a** Channel loss interval vs. success ratio. **b** Channel loss interval vs. redundancy

approaches to 100%, the redundancy increases exponentially confirming the theoretical findings in Section 6.

Analysis of channel loss interval impact: The objective of the simulations in this part is to analyze how we satisfy the required application success reliability α over a channel with different loss intervals and evaluate its cost.

In the simulations, we have a channel with uniform, triangular, and constant loss probabilities depending on the loss model. While triangular and uniform loss models use an interval, constant probability model uses the average of the interval as parameter. The initial loss interval is set to [0.02–0.08] and then we move this small loss interval over a larger value range [0.02–0.27] by 0.01 points towards right. In other words, we keep the variance fixed while incrementing the lower and upper bounds of the loss distribution. The number of application messages is 10 million and application demanded reliability is set to $\alpha = 0.98$ for each simulation.

Figure 2a and 2b show how the change in loss probability affects the empirically achieved α and its related cost. In both figures each loss interval is mapped to the midpoint of the interval on the x-axes and this midpoint serves as the expected value of both the triangular distribution with changing modes and the uniform distribution over the interval.

Figure 2a indicates that uniform probability channel loss with no loss estimation (UPNE), constant channel loss probability with perfect loss estimation (CPPE), and channel loss probability with regular (moving averages) estimation (TPRE) all attain the application required reliability $\alpha = 98\%$. The bottom overlapping line shows that in the long run, SSR-UDP packets can achieve the average success rate provided by the channel, i.e. expected value of the channel loss. Depending on the channel loss rate, 5% to 24% of network layer packets gets lost. However, the receiver always recovers 98% of the application layer messages.

UPNE loss model achieves 210% redundancy as presented in Fig. 2b. Remember that, UPNE does not employ any loss estimation approach but uses the highest loss rate achieved as the next channel loss probability and it shows the upper limits of redundancy. On the other extreme side, is CPPE which assumes that the channel loss is perfectly estimated. It draws the lower limit of redundancy between 19% and 73% over a channel which has average loss rate up to 24%. A more realistic case, TPRE, estimates the loss rate with moving averages and its redundancy is between 30–86%. Although 86% redundancy might look like too high, we should consider that it is obtained over a channel having 24% loss rate whilst achieving 98% reliability. Operating on a channel with less packet loss probability or setting the demanded application reliability to smaller values would reduce the redundancy. The waste is still around 4% because the number of batches generated is around 8% of the total application messages.

Analysis of channel loss interval length impact: In this part we show how success rate and redundancy gets affected with respect to UPNE and TPRE channel loss models with changing variance. In the simulations we set the mean of our channel loss to 0.12. Starting from a channel loss interval [0.11–0.13] while keeping the mean at 0.12, we increased the interval length up to 0.22. Since the variance effects our channel loss

estimation, in this experiment, we omitted CPPE channel loss probability model. The required channel reliability α is again set to 0.98 and 10 million application messages are generated for each simulation.

In Fig. 3 the x-axes shows the interval length rather than the middle point of the interval as in the previous experiment. Figure 3a shows that 98% reliability requirement has achieved by both UPNE and TPRE. Additionally, at the network layer both models sustain 88% packet success rate because average channel loss rate is 0.12 for both models. Triangular distribution with lower limit a, upper limit b, and fixed mode c has expected value $(a + b + c)/3$. However, since we constantly change the mode to the latest generated random value, its mean changes to $(a + b)/2$ in the results.

Figure 3b also shows that with UPNE the redundancy fluctuates between 120% and 110%. The reason behind fluctuation is the maximum loss rate that has been observed during the session. Yet, as the interval increases it is more probable to get a high loss rate for a batch and raise the amount of the redundant traffic put into the network. On the other hand, TPRE redundancy almost stays constant at 46%. Under TPRE loss model, changes in variance causes positive and negative channel loss estimations and in the long run these errors in estimations compensate the effect of each other and drag the observed metrics towards the average.

8 Conclusion and future work

Telesurgical Robot Systems (TRSs) is an important emerging cyber-physical technology in commercial and military domains. Currently, TRSs are mainly deployed on provisioned private wirelined networks with very limited impact on security or performance. However, the future deployment of them on various wireless networks including satellite and ad-hoc depends on the availability of efficient protocols to support security and reliability of TRSs.

Dealing with packet losses in TRSs requires forward error correction methods rather than packet retransmission due to the delay critical nature of TRS applications. Privacy concerns in domains such as military, requires utilization of the existing security metrics. However, applying an off-the-shelf crypto technique on FEC will introduce significant extra delay which is unacceptable for wireless TRSs. For example, even with a small value of k like 3 messages, AES shows an overhead increase from 41 to 61 operations for every 1% redundancy (59.8% average increase) with the FEC overhead. Our scheme uses information coding to encode a block of k application messages into n transmitted packets (where $n > k$) using random key-generated coefficients changing after each k application messages. The receiver will successfully decode the messages if at least k of them are received. The (n, k) is dynamically selected based on observed network conditions to maintain the security strength relative to the AES key length and to ensure packet loss and delay bounds as required by the application. The analytical proofs show that our privacy scheme can be as strong as AES with 128 and 256 key length, but with significantly less delay overhead (up to 40% when $k = 3$ and up to 28% when $k = 5$ for AES-128). In the reliability side, we have shown that even with high variation of packet loss, the protocol can achieve the target reliability threshold $\alpha = 99\%$ over a channel providing only 90% average reliability with a reasonable redundancy ranging between 30–45%.

In our future work, we plan to extend SSR-UDP with various packet loss models and test it on real world wireless networks.

References

1. SSR-UDP Simulator. Available at http://itom.utdallas.edu/. Accessed May 2011
2. Ahlswede R, Cai N, Li S-YR, Yeung RW (2000) Network information flow. IEEE Trans Inf Theory 46:1204–1216
3. Bakre A, Badrinath BR (1995) I-TCP: indirect TCP for mobile hosts. In: Liu J (ed) ICDCS 1995, 15th international conference on distributed computing systems. IEEE Computer Sciety, Los Alamitos, CA, pp 136–143
4. Balakrishnan H, Seshan S, Amir E, Katz RH (1995) Improving TCI/IP performance over wireless networks. In: MobiCom '95: proceedings of the 1st annual international conference on mobile computing and networking. ACM Press, New York, pp 2–11
5. Barsocchi P, Oligeri G, Potortì F (2006) Packet loss in TCP hybrid wireless networks. In: ASMS, May 2006
6. Bejczy AK (1980) Sensors, controls, and man-machine interface for advanced teleoperation. Science 208(4450):1327–1335
7. Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Crypto 96. LNCS 1109
8. Bellare M, Kilian J, Rogaway P (1994) The security of cipher block chaining. In: Advances in cryptology - Crypto 94. LNCS 839
9. Bolot JC (1993) End-to-end packet delay and loss behavior in the internet. In: SIGCOMM '93, vol 23. ACM, pp 289–298
10. Brady K, Tarn TJ (1998) Internet-based remote teleoperation. In: Proc. 1998 IEEE int. conf. robotics and automation, vol 1
11. Byers J, Luby M, Mitzenmacher M (2002) A digital fountain approach to asynchronous reliable multicast. IEEE JSAC 20(8):1528–1540
12. Chatzimisios P, Boucouvalas AC, Vitsas V (2004) Ieee 802.11 wlans: performance analysis in presence of bit errors. In: CSNDSP 2004

13. Coppersmith D, Winograd S (1990) Matrix multiplication via arithmetic progressions. J Symb Comput 9(3):251–280
14. Dai W (2010) Crypto++ library 5.6.0. http://www.cryptopp.com/
15. Dierks T, Rescorla E (2006) The transport layer security (tls) protocol, IETF RFC 4346. http://www.ietf.org/rfc/rfc4346.txt
16. Doarn CR, Anvari M, Low T, Broderick TJ (2009) Evaluation of teleoperated surgical robots in an enclosed undersea environment. Telemed J E Health 15(4):325–335
17. Gennaro R, Katz J, Krawczyk H, Rabin T (2010) Secure network coding over the integers. In: Proc. PKC. LNCS 6056, pp 142–160
18. Goertz RC, Thompson WM (1954) Electronically controlled manipulator. IEEE Commun Mag 12(11):46–47
19. Guthart GS, Salisbury JK (2000) The intuitive telesurgery system: overview and application. In: Proceedings of IEEE international conference on robotics and automation (ICRA), vol 1, pp 618–621
20. Harnett BM, Doarn CR, Rosen J, Hannaford B, Broderick TJ (2008) Evaluation of unmanned airborne vehicles and mobile robotic telesurgery in an extreme environment. Telemedicine and e-Health 14(6):539–544
21. Hill W, Green PS, Jensen JF, Gorfu Y, Shah AS (1994) Telepresence surgery demonstration system. In: Proceedings of international conference on robotics and automation, pp 2302–2307
22. Ho T, Koetter R, Medard M, Karger D, Effros M (2003) The benefits of coding over routing in a randomized setting. In: Proc. 2003 IEEE international symposium on information theory, p 442
23. Jaggi S, Sanders P, Chou PA, Effros M, Egner S, Jain K, Tolhuizen L (2005) Polynomial time algorithms for multicast network code construction. IEEE Trans Inf Theory 51:1973–1982
24. Kent S, Seo K (2005) Security architecture for the internet protocol, IETF RFC 4301. http://www.irtf.org/rfc/rfc4301.txt
25. King HH, Tadano K, Donlin R, Friedman D, Lum MJH, Asch V, Wang C, Kawashima K, Hannaford B (2009) Preliminary protocol for interoperable telesurgery. In: Advanced robotics 2009, pp 1–6
26. Koetter R, Medard M (2003) An algebraic approach to network coding. IEEE/ACM Trans Netw 11(5):782–795
27. Li SYR, Yeung RW, Cai N (2003) Linear network coding. IEEE Trans Inf Theory 49:371–381
28. Luby M (2002) LT codes. In: Proc. of the 43rd annual IEEE symposium on foundations of computer science (FOCS), pp 271–282
29. Lum M, Friedman D, King H, Donlin R, Sankaranarayanan G, Broderick T, Sinanan M, Rosen J, Hannaford B (2008) Teleoperation of a surgical robot via airborne wireless radio and transatlantic internet links. In: Field and service robotics, vol 42, pp 305–314
30. Lum MJH, Friedman DCW, Sankaranarayanan G, King H, Fodero K, Leuschke R, Hannaford B, Rosen J, Sinanan MN (2009) The raven: design and validation of a telesurgery system. Int J Rob Res 28:1183–1197
31. Nafaa A, Taleb T, Murphy L (2008) Forward error correction strategies for media streaming over wireless networks. IEEE Commun Mag 46(1):72–79
32. Padhye C, Christensen K, Moreno W, Christensen KJ (2000) A new adaptive fec loss control algorithm for voice over ip applications. In: Proceedings of IEEE international performance, computing and communication conference, pp 307–313
33. Plank J, Thomason M (2003) On the practical use of ldpc erasure codes for distributed storage applications. Technical Report UT-CS-03-510, University of Tennessee
34. Rescorla E, Modadugu N (2006) Datagram transport layer security, IETF RFC 4373. http://www.ietf.org/rfc/rfc4347.txt
35. Roychoudhuri L, Al-Shaer E (2010) Autonomic qos optimization of real-time internet audio using loss prediction and stochastic control. International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS) 1(3):81–107
36. Roychoudhuri L, Al-Shaer E (2005) Real-time packet loss prediction based on end-to-end delay variation. IEEE Transactions on Network and Service Management (IEEE TNSM) 2
37. Sankaranarayanan G, King H, Ko S, Lum MJH, Friedman DCW, Rosen J, Hannaford B (2007) Portable surgery master station for mobile robotic telesurgery. In: RoboComm '07. IEEE Press
38. Scott M (2010) Miracl: multiprecision integer and rational arithmetic c/c++ library. http://www.shamus.ie/
39. Shokrollahi A (2006) Raptor codes. IEEE Trans Inf Theory 52(6):2551–2567
40. Strassen V (1969) Gaussian elimination is not optimal. Numer Math 13:345–356
41. Tian Y, Xu K, Ansari N (2005) TCP in wireless environments: problems and solutions. Commun Mag, IEEE 43(3):S27–S32
42. Wang Y, Tozal ME, Al-Shaer E, Sarac K, Thuraisingham B, Chu B-T (2010) Information coding approach for secure and reliable telesurgery communications. Technical Report UNCC-SIS-10-7-1. http://www.cyberdna.uncc.edu/publications.php