On Secure and Resilient Telesurgery Communications over Unreliable Networks

M. Engin Tozal[+], Yongge Wang[*], Ehab Al-Shaer[*], Kamil Sarac[+], Bhavani Thuraisingham[+], Bei-Tseng Chu[*]
[+]Department of Computer Science, The University of Texas at Dallas
[*]Department of Software and Information Systems, University of North Carolina Charlotte
{engintozal,ksarac,bxt043000}@utdallas.edu, {yongge.wang,ealshaer,billchu}@uncc.edu

***Abstract*—Telesurgical Robot Systems (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 light-weight 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 other similar cyber-physical wireless application domains.**

I. Introduction

Telesurgical Robot Systems (TRSs) have been recently the focus of research in academic, military, and commercial domains [14]. 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 [13]. 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, design and implementation of a secure and reliable wireless communication conforming to the application tolerable delay for TRSs is crucial.

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 [12]. 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 [12]. 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.

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 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 adaptive FEC obtains unacceptable delay for wireless TRSs as the AES encryption delay overhead increases significantly with the traffic redundancy (see Section VI-A). In addition, many of the existing FEC methods based on digital fountain [7], [16] require the content of the messages to be known before the communication session and the FEC methods based on network coding [1] are tailored for multicast communication rather than unicast.

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 reliability υ such that $0 < \upsilon < \alpha < 1$. 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, TLS/SSL, and IPSec. 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.

The rest of the paper is organized as follows: Section II presents the related work. The proposed information coding technique, cryptographic functions and their analysis is introduced in Sections III and IV respectively. Section V presents dynamic adjustment of message encoding based on the observed loss and delay in the channel. We discuss the experimental results in Section VI and present conclusions and future works in Section VII.

II. Related Work

Since the first telesurgical robot system [9] which was a couple of mechanical hands cabled to a remote handle, many successful works have been done in the field [3], [10], [11]. Brady and Tarn [6], developed a framework for extending teleoperation systems to Internet. Project RAVEN [13] is an implementation of TRSs and it proved that remote surgeries can successfully be realized over the Internet via UDP.

Lum et. al. [13], 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 using an unmanned airborne vehicle as a network topology.

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 (FEC). Many FEC techniques based on digital fountain have been designed in the past [7], [16] 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, (i) the message is required to be known before the communication starts and (ii) privacy protection is not addressed. Network coding is also used for multicast communication. 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 (i) it requires the deployment of network coding capability into intermediate nodes, (ii) it is suitable for multicast communication rather than unicast, and (iii) privacy protection is not addressed.

III. Information Coding

In the traditional Hill cipher scheme, one encrypts an alphabet sequence $(m_1, m_2, \ldots, m_n)$ with an $n \times n$ key matrix A by letting the cipher text $(c_1, c_2, \cdots, c_n)^T = A(m_1, m_2, \ldots, m_n)^T \bmod 26$. We can extend Hill cipher to an extended coding scheme to achieve reliability and privacy at the same time. Specifically, our 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 $x_1, x_2, \ldots, x_q$.

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 V.

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 another word, 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 $1 \le i \le n$, $1 \le j \le k$. 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 [4]) for each of these n messages, and deliver these n encoded messages together with their authentication tags to the receiver independently.

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 uncorrupted packets (any k packets will be OK and the order is not important) from the collection of these paths. 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 another word, if the UDP packet loss is at most 25%, we have achieved reliable communication channels using the UDP protocol.

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 $F_{2^{160}}$). This will improve the performance by a factor of at least 20. In particular, the decoding coefficients matrix in the equation (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.

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.

IV. Secure Key Generation Functions

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.

V. Dynamic Adjustment of (n, k)

The performance of the proposed 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, 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.

A. 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 a sender and a 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 [2], [8]. 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:

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

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., $P\{\textit{Successful Batch}\} = \alpha$. 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 (3), we can work with the expression $P\{\textit{Failed Batch}\} = 1 - \alpha$.

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

Solving (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, (4) does not have a closed form solution. To develop a "numerical"

solution, we resorted to normal approximation[1] 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

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

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

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

Given that k and α are constants in (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.

B. 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 (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: (i) real-time data have an application-dependent delay threshold and increasing k by accumulating more messages will likely to increase the delay, and (ii) 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 achieve RSA-1024 level of security we require $kn \geq 10$. 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 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

Require: k { current value of k }
Require: α { application reliability requirement }
Require: p_{next} { estimated packet success probability for the next batch }
Require: p_{last} { estimated packet success probability for the previous (last) batch }
Require: Δp_0 { previous amount of change in packet success probability }
Require: d_{next} { estimated channel delay for the next batch }
Require: d_{max} { maximum tolerable real time application delay }
Require: d_{imsg} { delay between generation of two application messages }
Ensure: 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 Equation 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. $\mathcal{E}_1$ at line 5 implies occurence of a small amount of change in the estimated packet loss probability. 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

[1]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.

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 term $\mathcal{E}_2$ is a very small amount used to increase the estimated channel delay to compensate with estimation errors. 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 Equation 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 [2], [5]. Nevertheless, the error in these estimations affect optimization of k and n.

VI. Performance and Simulation Comparisons

A. 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.

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 equations (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 equation (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 precompute 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 precomputations, 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-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 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}.

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 [15], 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}.

TABLE I: 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	Sender	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

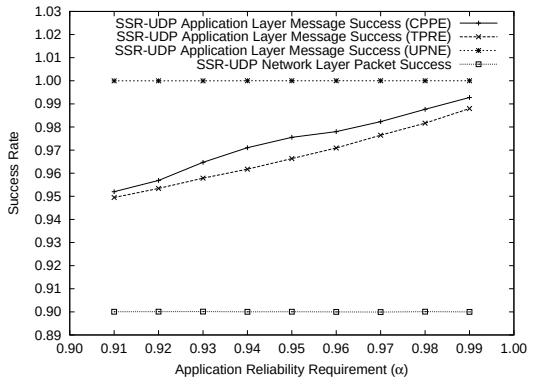
Finally, in Table I, we list the comparison data for several examples with the above assumption using various values of k (3, 5 and 7) that obtain a reasonable delay bound (less than 150ms one-way), and various values of n that represent up to 50% redundancy.

B. Evaluation of Secure and Statistically Reliable UDP

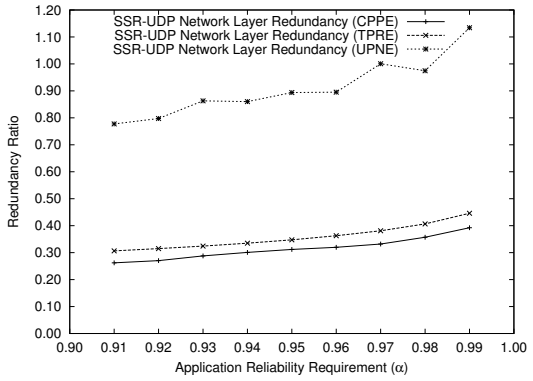
In this section we present our simulation-based evaluation of SSR-UDP using a custom built simulator. Our evaluation metrics include reliability, redundancy, and waste and our parameters are application demanded reliability (α), channel loss distribution, and channel loss variance. Our experiments include 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 loss rate observed 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. Although our worst case (UPNE) and best 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.

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.

Figures 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 Figure 1a SSR-UDP achieves at least the required reliability α under all loss models. The top line in Figure 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 channel loss,



(a) Application Reliability Requirement (α) vs. Success Ratio



(b) Application Reliability Requirement (α) vs. Redundancy

Fig. 1: Behavior of Success Rate and Redundancy with respect to Application Reliability Requirement (α)

it achieves (very close) to 100% reliability at each application demanded reliability α. On the other hand, Figure 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 Figure 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 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.

Additionally, we ran a set of simulations to see how redundancy changes with a relatively high level reliability requirement [17]. We achieved 99.9% success rate with 79% redundancy for the above TPRE loss model. We also observed that as the reliability requirement approaches to 100%, the redundancy increases exponentially confirming the theoretical findings in Section V.

Finally, in another set of simulations [17], we fixed application required reliability α at 98% and observed the attained reliability as well as redundancy over channels having various loss rates between 2% and 24% again under UPNE, TPRE, and CPPE models. Our results show that SSR-UDP successfully achieves the required reliability over all channels with redundancy changing between 30% to 86% for the more realistic TPRE case.

VII. Conclusions

Dealing with packet losses in TRSs requires FEC 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 unacceptable by wireless TRSs. Our scheme uses information coding to encode a block of k application messages into n transmitted packets where $n > k$. 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. 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). We have also 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.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, 2000.

[2] P. Barsocchi, G. Oligeri, and F. Potortì. Packet loss in TCP hybrid wireless networks. In *ASMS*, May 2006.

[3] A. K. Bejczy. Sensors, Controls, and Man-Machine Interface for Advanced Teleoperation. *Science*, (208):1327–1335, June 1980.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Crypto 96, LNCS 1109*, 1996.

[5] J.C. Bolot. End-to-end packet delay and loss behavior in the internet. In *SIGCOMM '93*, volume 23, pages 289–298. ACM, October 1993.

[6] K. Brady and T.J. Tarn. Internet-based remote teleoperation. *Proc. 1998 IEEE Int. Conf. Robotics and Automation*, vol.1, 1998.

[7] J. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE JSAC*, 20(8):1528–1540, 2002.

[8] P. Chatzimisios, A. C. Boucouvalas, and V. Vitsas. Ieee 802.11 wlans: performance analysis in presence of bit errors. In *CSNDSP 2004*, 2004.

[9] R.C. Goertz and W.M. Thompson. Electronically Controlled Manipulator. *Communications Magazine, IEEE*, November 1954.

[10] G.S. Guthart and J.K. Salisbury. The Intuitive Telesurgery System: Overview and Application. April 2000.

[11] W. Hill, P.S. Green, J.F. Jensen, Y. Gorfu, and A.S. Shah. Telepresence Surgery Demonstration System. May 1994.

[12] H. H. King, K. Tadano, R. Donlin, D. Friedman, M.J.H. Lum, V. Asch, C. Wang, K. Kawashima, and B. Hannaford. Preliminary protocol for interoperable telesurgery. In *Advanced Robotics 2009*, pages 1–6,, 2009.

[13] M.J.H. Lum, D.C.W. Friedman, G. Sankaranarayanan, H. King, K. Fodero, R. Leuschke, B. Hannaford, J. Rosen, and M.N. Sinanan. The raven: Design and validation of a telesurgery system. *Int. J. Rob. Res.*, 28:1183–1197, September 2009.

[14] G. Sankaranarayanan, H. King, S. Ko, M.J.H. Lum, D.C.W. Friedman, J. Rosen, and B. Hannaford. Portable surgery master station for mobile robotic telesurgery. In *RoboComm '07*, pages 1–8. IEEE Press.

[15] M. Scott. Miracl: Multiprecision integer and rational arithmetic c/c++ library. http://www.shamus.ie/, 2010.

[16] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[17] Yongge Wang, M. Engin Tozal, Ehab Al-Shaer, Kamil Sarac, Bhavani Thuraisingham, and Bei-Tseng Chu. Information coding approach for secure and reliable telesurgery communications. Technical Report UNCC-SIS-10-7-1, http://www.cyberdna.uncc.edu/publications.php, July 2010.