Smart replication for seamless and efficient real time communication in underground railway train

Kariem Fahmi, Douglas Leith
Trinity College Dublin, Ireland

***Abstract*—In this paper, we consider the task of transporting mission-critical traffic, with strict QoS requirements for high availability and low packet loss, in underground trains that are equipped with multiple WiFi backhauls. We observe that the commonly used simple replication approach for mitigating frequent handovers comes at the cost of many duplicate packet transmissions and so wasted network capacity. Instead we take the approach of predicting when a handover is about to occur and only replicating packets shortly before it happens, significantly reducing the amount of redundancy needed and freeing up network resources. We propose a novel neural network predictor, based only on features extracted from the WiFi client log, that predicts when a handover will occur. We train and evaluate our predictor on a large data set collected in a production environment, and show it predicts handover with high accuracy. We use this predictor as a basis for an adaptive replication scheme and compare it against simple replication and no replication. Results show that when application throughput is low compared to link capacity, our scheme achieves 91% of the loss reduction provided by simple replication on average while using only 13% of the redundancy and experiencing 20% less latency. When application throughput is high compared to link capacity, simple replication causes more more losses compared to no replication, while our scheme maintains a lower loss rate compared to no replication.**

I. Introduction

In this paper we consider the setup shown schematically in Figures 1 and 2. A subway train is equipped with two WiFi clients, one at the front and one at the rear of the train. WiFi access points (APs) are placed along the trackside. As the train moves along the track the WiFi clients switch between APs. The network connection of a client is interrupted when it switches between two APs, but by placing the trackside APs less than one train-length apart the handovers of the two clients are de-synchronised i.e. at all times at least one WiFi client on train is connected to an AP, see Figure 1(b). When a client switches between APs downlink packets queued at the old AP are discarded, as are any uplink packets that exceed their delivery deadline due to the delay caused by switching. Packet loss can therefore occur at each handover, as illustrated in Figure 2(b). The network capacity available to the train also fluctuates greatly over time since sometimes both clients are connected and sometimes only one client, plus the bandwidth of the wireless link(s) is variable as the train moves and the external environment changes.

Our interest is in managing the quality of service at the transport layer using an over-the-top approach. Traffic to and from the client stations on the train is routed via a proxy on the wired network close to the APs, Figure 2(a). This creates the freedom to implement new transport layer behavior over the path between proxy and clients. The approach assumes only control of the proxy client and gateway, but no knowledge or control of the internal workings of the WiFi clients or the APs. The great advantage of this over-the-top approach is its ease of rollout since no changes are required to existing AP's or WiFi clients. However, the price is that the internal WiFi decision-making on when to switch between APs is hidden i.e. we can observe when switching occurs but we cannot observe or change the switching logic.

The WiFi connection between a subway train and the trackside carries two main classes of traffic: (i) train control-

Email: kfahmi@tcd.ie, doug.leith@tcd.ie

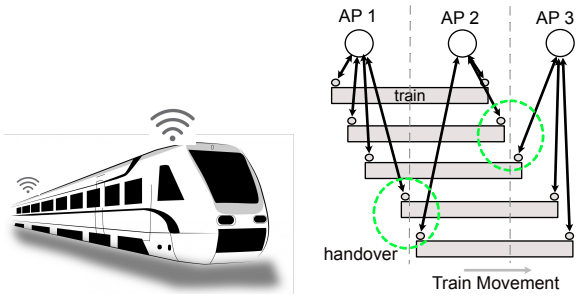


Fig. 1: Illustrating setup considered. (a) A train is equipped with two 802.11ac WiFi clients, one at the front and one at the rear. WiFi access points are installed along the trackside. (b) The spacing of the trackside access points is selected so that at least one of the train WiFi clients is connected at any time, e.g. for a train of 150m length the trackside access points might be spaced 100m apart.

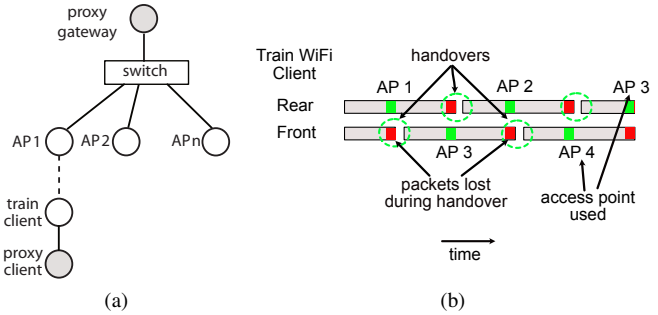


Fig. 2: (a) Edge proxy setup. (b) Illustrating downlink packet transmissions from the train front and rear WiFi clients. As the train moves the AP to which each client is connected changes, e.g. the rear WiFi client is initially connected to AP1, then to AP2 and so on. During handovers there is a break in connectivity and packets queued at the old access point are lost (indicated in red).

plane traffic (i.e. Communication Based Train Control (CBTC) traffic on driver-less trains, CCTV monitoring for track and on-board video surveillance, mission-critical VOIP) and (ii) passenger data traffic from public WiFi hotspots within the train. Here we focus on (i), where the QoS requirement is for high availability, and low packet loss [1].

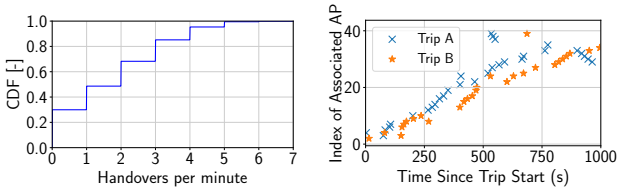
Since packets may be lost at each handover, multiple studies have proposed to replicate traffic to multiple wireless clients as a way of ensuring quality of service in train-to-ground (T2G) [2] [3] [4]. Since one client is, by design, always connected this avoids packet loss. However, this comes at the cost of many duplicate packet transmissions and so wasted network capacity.

A smarter approach is to predict when a handover is about to occur and then only replicate packets during this short period, illustrated by the green shading in Figure 2(b), and it is this approach that we study here. By reducing the number of duplicate packets sent the potential exists to free up significant amounts of network capacity for more productive uses, e.g. for carrying passenger data traffic. It is important to stress that our goal is to *predict* when a handover is about to occur and not to trigger the handover since, as already noted, we have no internal access to the WiFi APs/clients and so have no control over handovers.

It turns out that accurate prediction of handovers is surprisingly tricky since, based on extensive measurements from trains operating in a production environment, the WiFi APs/clients exhibit complex behaviour. The sequence of handovers can change substantially between runs over the same track and predictions must take account of periods when the train is stationary (such as when waiting in a station or stopped at a signal), when the train is accelerating/decelerating etc. We propose a novel neural network-based predictor to address these issues and evaluate its performance using experimental data.

A key observation is that a fundamental trade-off exists between the accuracy of the handover prediction, the level of packet replication and the level of packet loss. Namely, false positives (a handover is predicted but does not actually occur) cause unnecessary packet replication whereas false negatives (it is predicted that there will not be a handover, but one occurs) lead to unnecessary packet loss. We can always avoid false negatives trivially by always predicting that a handover is about to occur, and this yields the wasteful strategy of continuously replicating packets to both WiFi clients. We can also avoid false positives by always predicting that no handover will occur, and this yields the strategy of never replicating packets and so results in a high loss rate. Our second main contribution is to charactertise the middle ground between these two extremes and quantify the trade-offs.

We propose a simple adaptive packet replication scheme and evaluate its performance. Results show that when application throughput is low compared to link capacity (i.e. extra redundancy does not come at the cost of increased congestion losses), our scheme achieves 91% of the loss reduction provided by simple replication on average while using only 13% of the redundancy and experiencing 20% less latency. However, when application throughput is high compared to link capacity, simple replication causes more more losses compared to no replication making it counter productive, while our scheme maintains a lower loss rate compared to no replication.



(a) (b)

Fig. 3: Experimental measurements of train-to-trackside WiFi handover events on an underground train.

In summary, our main contributions include: (i) development of a neural network-based handover predictor, (ii) quantifying the trade-offs between prediction accuracy, packet replication and loss, (iii) development of a simple adaptive packet replication scheme and its performance evaluation using experimental data.

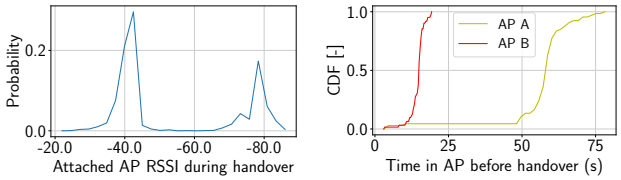
II. How Do T2G WiFi Handovers Behave?

We collected the logs generated by the on-board WLAN client on an underground train using a commercial T2G system during 33 trips over the same track, spanning over 17 hours. The client logged the result of all RSSI scans performed during the trip as a list of MAC addresses with corresponding RSSI values and the current time. It also logged each handover event, recording the MAC address of the newly associated AP and the current time. By analyzing the logs, we were able to parse 58935 RSSI scans and 2455 handover events.

Fig. 3(a) shows the measured CDF of the number of handovers per minute i.e. time is divided into one minute slots and the empirical CDF of handovers per slot calculated. Observe that in around 30% of time slots no handovers occur. Note that this data includes time slots where the train is stopped in a station and at signal lights and so when the train is moving the fraction of slots with no handover is significantly smaller. It can also be seen that around 60% of slots experience between 1 and 3 handovers and around 15% of slots experience 4-7 handovers. The handover duration i.e. of the time between disassociating from the old AP and associating with the new AP takes at least 40ms and in 99% of handovers take no more than 200ms. During this time network connectivity is lost and following handover packets queued at the old AP are discarded[1]

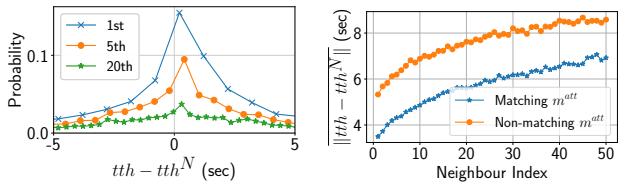
Fig. 3(b) shows the APs that were connected to during two different trips across the same section of track. Observe that the sequence of APs differs significantly despite the fact that these measurements are for the same section of track. While some of the changes are essentially a shift/offset in the sequence, presumably due to fluctuations in train speed, more complex differences are also evident. For example, APs used on one trip need not be used on the second trip and the order in which APs are connected can be reversed between trips (see the right-hand side of the plot, around time 800-900s).

[1]There currently exists no WLAN standard to facilitate forwarding data during a handover. While there were previous attempts to create such a standard, namely the Inter-Access Point Protocol (IAPP) which was referred to as 802.11F, they were not successful and the standard has since been withdrawn [5].



(a) (b)

Fig. 4: Relationship between handover and the RSSI of the attached AP and the time spent connected.



(a) (b)

Fig. 5: Fig (a) shows the PDF of the difference between the time till handover (TTH) associated with a scan and the TTH associated with similar scans/neighbours sorted by Euclidean similarity. The plot shows the PDF for 3 neighbor values: 1, 5 and 20. Fig (b) shows the mean of the same value across different neighbor indexes, both when limiting the comparison space to scans that share the same attached AP, matching m^{att}, and when no limits were placed, non-matching m^{att}.

To try to gain more insight into the factors that affect the complex handover behaviour, Figure 4(a) plots the empirical PDF of the RSSI of the AP to which the WLAN client is attached when a handover starts. It can be seen that that handovers are concentrated around two ranges of RSSI values, -40dB and -80dB. It is not surprising that a low RSSI of -80dB increases the probability of a handover since this likely another AP has higher RSSI and so would be preferred. However, Figure 4(a) also indicates that a much higher RSSI of -40dB is also likely to be associated with a handover. The reason for this is not clear, presumably it is due to interactions between the complex radio environment and the handover switching logic within the APs/clients, but it is clearly a useful observation for predicting when a handover may occur.

For two different APs Fig 4(b) shows the empirical CDF of the time spent attached to the AP before handover occurs. When connected to AP B the train is generally stopped in a station and so the time spent attached to the AP is rather long (> 1 minute), whereas the train is generally moving when connected to AP A and so the train spends a much shorter time attached to AP A (around 12 seconds). While the overall time spent attached to an AP is therefore highly variable, the minimum time is around 10 seconds and this again can be used to assist with predicting when a handover may occur.

III. Predicting Handovers

We would like to predict whether a handover will occur on a train WLAN link within α seconds. Here α is a design parameter that reflects the amount of time needed to flush the associated AP's buffer before an upcoming handover and to switch to transmitting packets primarily over the second train WLAN link (perhaps with duplicates still sent over the link that is due to handover). We select α to be 3 seconds. We approach this prediction task as a binary classification problem. Each time a train WiFi client carries out a scan for nearby APs we construct a feature vector X, input this to a classifier which outputs +1 if a handover is predicted to occur with the next α seconds and otherwise outputs -1. Since the train traverses the same route every day, measurements of handovers are readily available for use as training data for the classifier.

A. Feature Selection

1) Baseline Features: Based on the data in Figure 4, plus our understanding of the wireless system, it is clear that the RSSI of the AP to which a client is attached and the time since the last handover are likely to be useful features for predicting handover. Also we use the MAC address of the attached AP as a feature since we expect there may be consistent differences between APs due to differences, for example, in the local radio environment. Since we know that the train changes speed and, in particular, can spend extended periods time stationary we also include the rate of change of the RSSI as a rough proxy for train speed.

More formally, let $r^{att}_{i,j}$ denote the RSSI of the attached AP during scan i in trip j and $tsh_{i,j} = t_{i,j} - t^{PHO}_{i,j}$ be the time since the last handover, where $t_{i,j}$ is the time of scan i in trip j and $t^{PHO}_{i,j}$ is the time of the last handover before the scan. We encode the AP MAC address using one-hot encoding i.e. as a vector of length n, where n is the number of APs, and with all elements 0 except for the element corresponding to the currently attached AP which has value 1, and is referred to as $\vec{M}^{att}_{i,j}$. Letting $M_{i,j}$ be the set of MAC addresses observed in scan i then the rate of change of RSSI is $s_{i,j} = \sum_{m \in M_{i,j} \cap M_{i-1,j}} \|(r^m_{i,j} - r^m_{i-1,j})\|$ where $r^m_{i,j}$ is the RSSI of the AP with MAC address m.

2) TTH of nearest neighbours: In addition to the baseline features, we use the time till handover (TTH) associated with scans that have a similar RSSI fingerprint to the current scan as an additional feature. The underlying intuition is that similar RSSI fingerprints will occur in similar physical locations and thus the distribution of their TTH should be similar. Fig 5(a) shows the PDF of the difference between the time till handover (TTH) associated with a scan and the TTH associated with similar scans/neighbours sorted by Euclidean similarity. As shown, for the closest neighbour (i.e. 1st) , there is roughly 15% chance of no error at all and a high probability of low error, making it reasonable estimate. Since variations in speed and noise in similarity, caused by fluctuations in RSSI, will contribute error to this estimate, the TTH from the top k similar scans is used , which allows the classifier to assign appropriate weights to them. Additionally, the probability of larger difference/error increases as the index of the neighbor index gets higher, making the choice of k important. We have empirically selected the value of k to be 15 in our implementation.

More formally, let $tth_{i,j} = t^{NHO}_{i,j} - t_{i,j}$ denote the TTH from scan i in trip j, where $t_{i,j}$ is the timestamp of the scan and $t^{NHO}_{i,j}$ is the timestamp of the next handover after that scan. Having defined the tth for all scans, we compute the vector of k tth estimates for scan i, j,

$\vec{TTH}_{i,j}$=$[tth_1, tth_2 \dots tth_k]$, where each tth estimate is associated with the respective RSSI scan in the vector of k most similair RSSI scans, $\vec{N}_{i,j}$=$[\vec{R}_1, \vec{R}_2 \dots \vec{R}_k]$, such that tth_a is the time till handover after RSSI scan $\vec{R}_a$. An RSSI scan $\vec{R}_a$ is the vector of RSSI values appearing in scan a. $\vec{N}_{i,j}$ is sorted using similarity function D, such that $D(\vec{R}_{i,j}, \vec{R}_a) \leq D(\vec{R}_{i,j}, \vec{R}_{a+1})$. In our implementation we select D to be Euclidean Similarity. The similarity between any two scans a and b is calculated as follows

$$D(R_a, R_b) = \sqrt{\sum_{m \in M_a \cup M_b} (r_a^m - r_b^m)^2} \tag{1}$$

where M_a and M_b are the set of MAC addresses appearing in scan a and scan b respectively, and r_a^m and r_b^m are the RSSI value of AP with MAC address m in scan a and RSSI value of AP with MAC address m in scan b respectively. In cases that $M_a \neq M_b$ (i.e. an AP appears in scan a but not in scan b or vice versa) a sentinel value is assigned for the RSSI of any missing MAC in a scan, such that $r_a^m = -90$ if $m \notin M_b$ and similarly $r_b^m = -90$ if $m \notin M_a$.

To improve accuracy of the tth estimates in $\vec{TTH}_{i,j}$, the comparison space of a scan i, j is limited to the set of scans $C_{i,j}$, defined below, where the currently attached APs match

$$C_{i,j} = \{R_a \quad | \quad m_{i,j}^{att} = m_a^{att}\} \tag{2}$$

First, this greatly reduces the number of scans in the comparison space, improving the time complexity. Second, it also improves the accuracy as it avoids the problem of assigning high similarity to two scans where one occur just before a handover and one that occurs just after, and thus have both have a similar RSSI fingerprint but a very different tth. 5(b) shows the relation between the index of the neighbor on the x-axis and the mean difference between the original scan and the neighboring scan on the x-axis, both when limiting the comparison space to scans that share the same attached AP, Matching m^{att}, and when no limits were placed, Non-matching m^{att}. We can clearly observe that the difference was higher when the attached AP wasn't matched.

B. Feature Engineering

1) Normalization: All the input features, except $m_{i,j}^{att}$, are continuous and have different scales and are normalized as follows

$$z_{i,j} = \frac{x_{i,j} - \bar{x}_j}{s_{x_j}} \tag{3}$$

where $x_{i,j}$ is the ith value of the feature j, $\bar{x}_j$ and s_{x_j} are the mean and standard deviation of the feature over the training dataset, respectively.

2) Random oversampling of handover events: Since only 4% of the scans are handover events, this results in a severe imbalance in the number of class labels and may cause the cost function of the classifier to be skewed towards predicting non-handovers. To address this problem, random oversampling of handover events is performed to increase the number 25-fold, matching the number of non-handovers.

C. Training and prediction

Since this is a binary classification problem, the classifier is trained using the input feature vector $\vec{X}_{i,j}$, corresponding to features extracted from scan i in trip j, and the binary output class label $y_{i,j}$ defined as follows

$$y_{i,j} = \begin{cases} 1, & tth_{i,j} \leq \alpha \\ 0, & tth_{i,j} > \alpha \end{cases} \tag{4}$$

For prediction, the class probability generated from the classifier, $P(\hat{y}_{i,j} = T|\vec{X}_{i,j})$, is compared against a threshold θ in order to predict the handover as follows

$$\hat{y}_{i,j} = \begin{cases} 1, & P(\hat{y}_{i,j} = T|\vec{X}_{i,j}) \geq \theta \\ 0, & P(\hat{y}_{i,j} = T|\vec{X}_{i,j}) < \theta \end{cases} \tag{5}$$

D. Evaluation metric

The evaluation metric is based on the number of true positive (TP), which indicate correctly identified handover events, and false positives (FP), which indicate incorrectly identified non-handover events. Bad performance on TP will result in losses, while bad performance on FP will result in wasted capacity. Since the dataset is highly biased towards non-handover events, we can't rely on accuracy for measuring the performance of a classifier. Instead, the performance is measured as the area under the curve (AUC) of the receiver operating characteristics (ROC) curve. To generate the ROC curve, 33-fold cross validation is performed with 100 values for θ between 0 and 1, and the mean number of TP and FP is calculated for each value and represents a point in the ROC curve. The rationale for picking 33 folds is that each testing fold consists of data from one trip, and the rest of the trips are used for training. This mimics reality, where data from all previous trips is available for training the model and the resulting model is then tested on the current trip. The mean number of TP over all testing folds is used in the ROC curve and is calculated as follows

$$\overline{TP} = \frac{\sum_{i=0}^{N_{trips}} \frac{TP_i}{TP_i + FP_i}}{N_{trips} - 1} \tag{6}$$

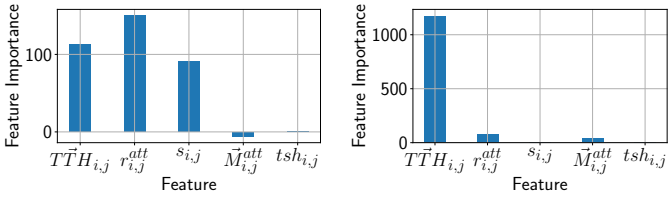
where TP_i and FP_i are the TP and FP from the iteration of the cross-validation over trip i and N_{trips} is the total number of trips in the dataset. The mean number of FP is calculated in a similar manner

$$\overline{FP} = \frac{\sum_{i=0}^{N_{trips}} \frac{FP_i}{TP_i + FP_i}}{N_{trips} - 1} \tag{7}$$

E. Classifiers

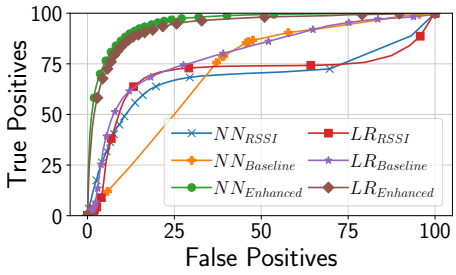
We tested 2 learning approaches for handover prediction: artificial neural network (ANN) and logistic regression (LR). Each approach was tested with three different set of features:

- RSSI-only (C_{RSSI}) – This feature set includes only the RSSI of the currently attached AP i.e. $\vec{X}_{i,j} = [r_{i,j}^{att}]$.
- Baseline features ($C_{Baseline}$) – This set includes the baseline features without any feature engineering i.e. $\vec{X}_{i,j} = [r_{i,j}^{att}, tsh_{i,j}, \vec{M}_{i,j}^{att}, s_{i,j}]$
- Enhanced features ($C_{Enhanced}$) – This is an enhanced feature list with the feature engineering steps including



(a) Artificial Neural Network (b) Logistic Regression

Fig. 6: Figure shows feature importance score for (a) Artificial Neural Network and (b) Logistic Regression calculated using the feature importance permutation approach



(a)

Fig. 7: ROC curve showing false positives and true positives at different values for the decision threshold, θ, for three Artificial Neural Network classifiers NN_{RSSI}, NN_{BL} and NN_{Ext} trained on the RSSI only feature set, the baseline feature set and the extended feature set respectively, and three Logistic Regression classifiers LR_{RSSI}, LR_{BL} and LR_{Ext} trained on the same three feature sets.

the normalization and random sampling. Additionally, features that are un-important for the specific classifiers are removed using feature importance permutation, which is described in more detail below.

1) Feature selection based on feature importance: In order to eliminate features that are either useless or have a negative impact on the performance of the classifiers, the importance of each feature is measured using the permutation feature importance method and un-important features are removed from the final set. To do this, the classifier is trained on a multiple versions of the training dataset where each version has the values for one feature randomly shuffled. The performance of each classifier version is compared to the performance on the normal classifier (i.e. without any features shuffled) and if the permuted version produces worse performance, then the feature is important, otherwise it is not. To measure performance of each classifier version, the AUC of the ROC curve is used. Fig 6 shows the feature importance score for the ANN and the LR. Based on the results, the extended feature set for ANN will include all features except $\vec{M}^{att}_{i,j}$ and $s_{i,j}$ i.e. $\vec{X}_{i,j} = [r^{att}_{i,j}, tsh_{i,j}, T\vec{T}H_{i,j}]$, while the feature set for the LR will include all features except $r^{att}_{i,j}$ and $s_{i,j}$ i.e. $\vec{X}_{i,j} = [tsh_{i,j}, \vec{M}_{i,j}, T\vec{T}H_{i,j}]$

2) Hyper parameter tuning for ANN: The ANN is configured to use the logistic activation function, ADAM Solver, cross-entropy cost function, L2 regularization with a coefficient of 0.1, and one hidden layer with 20 nodes. The configuration values were selected using the k-fold cross validation scheme discussed in section III-D.

3) Hyper parameter tuning for LR: The LR is configured to use the Limited-memory BFGS solver and L2 regularization with a 0.2 coefficient. The configuration values were selected using the k-fold cross validation scheme discussed in section III-D.

F. Evaluation

1) Implementation: The evaluation was implemented in python using Sickit learn.

2) Results: Fig 7 shows the ROC curve resulting from evaluating the 6 classifiers described in the previous section at 100 different values of θ. We can see that the RSSI only classifiers, LR_{RSSI} and NN_{RSSI}, performed the worst and were fairly close in performance. The NN baseline classifier, NN_{BL}, performed worse than the LR baseline classifier, LR_{BL}, as it may have been more affected by the imbalance of the training dataset. The two best classifiers, LR_{Ext} and NN_{Ext}, performed much better than the rest and were fairly close to each other, with the NN_{Ext} having slightly better performance. Based on the results, we will use the NN_{Ext} classifier in the implementation of the smart replication system.

IV. Smart Replication System

The proposed Smart Replication system combines multi-path replication based on handover prediction and multi-path load balancing in order to minimize handover losses using the least possible redundancy while fully utilizing the leftover capacity for non-redundant traffic.

Figure 8 shows the system architecture.

A. Transparent tunneling

The system uses a VPN tunnel in the on-board gateway and the network-side gateway in order to transparently re-route all traffic exchange between the train and the ground through a user-space process that implements the logic of the smart replication system. As shown in Figure 8, in the on-board gateway the VPN client creates a virtual tunnel interface (TUN) and configures the routing such that all incoming traffic is routed through this interface. The VPN server does the same on the network-side gateway.

B. Protocol

The system uses special protocol headers to perform the necessary signaling for connection establishment, subflow addition, data exchange and handover signaling from the VPN client to the VPN server. All traffic between the VPN server and the VPN client is encapsulated inside this protocol. The header structure for the protocol is shown in Fig. 9.

a) Connection establishment: When the VPN client is started, it will establish a connection with the VPN server. This is done using a traditional TCP-like handshake based on SYN, SYN-ACK and ACK. The server will respond to the SYN packet with a SYN-ACK, and the client will respond to the SYN-ACK with an ACK. The sequence number field is assigned a random value by the client in the SYN packet and by the server during the SYN-ACK packet.

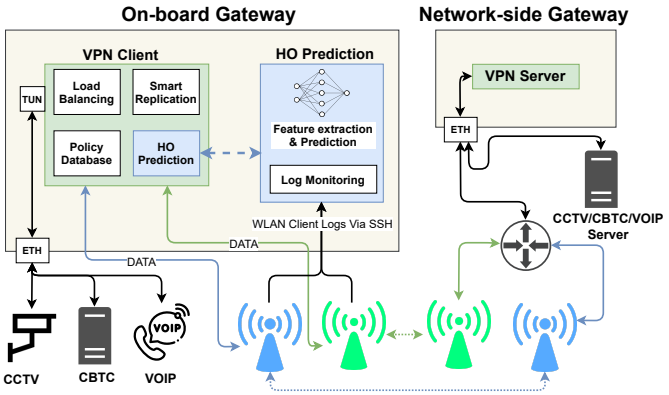


Fig. 8: Smart Replication System Architecture

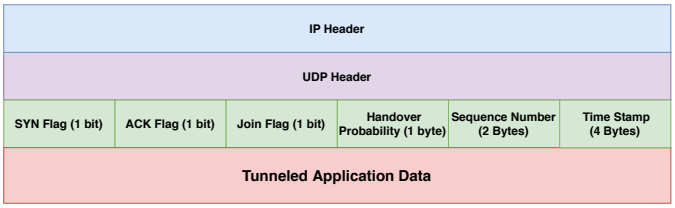


Fig. 9: VPN Protocol Header Structure

b) Subflow Addition: The SYN-ACK packet sent to the client includes a special token used to identify future subflows attempting to join the original connection. The algorithm for generating the token is identical to the MPTCP token generation process [6]. Once the handshake concludes, the client will begin adding subflows by transmitting a SYN-JOIN packet, containing the token received previously, from other WiFi links. Upon receipt of the SYN-JOIN, the server will map the joined subflow to the original session and will utilize it for transmitting data. It will also transmit a SYN-JOIN-ACK over the new subflow as a response, and the client will respond to this with an ACK.

c) Data Exchange: Once a connection is established between the VPN client and server, layer 3 packets from applications will be tunneled under the protocol header. For each packet received in the VPN client or server, a sequence number counter is incremented and assigned to that packet. The purpose of the sequence number is to identify replicated packets at the receiving end to avoid delivering the same packet multiple times to the application.

C. Load Balancing

The load balancing module assigns application flows to one of the available WiFi links. Each application flow is identified by the IP 5-tuple (source IP, destination IP, source port, destination port and transport protocol). The link assignment is done in a weighted fashion with each class of application traffic having its unique weight. A weight reflects an application's expected throughput. For example VOIP, CCTV and CBTC ar three different traffic classes, with CCTV having the highest weight since it has the highest expected throughput. A flow is associated with a traffic class by consulting the policy database, which contains mappings from any combination of the IP 5-tuple to a weight. The goal of the load balancer is to distribute the flows across the links to minimize the discrepancy in weight assigned between links. This will result in approximately the same amount of data being transmitted over each link.

The assumptions behind the load balancing approach are two fold:

- all the on-board WiFi links will experience approximately the same capacity profile over the duration of the trip since they traverse the same space and associate with similar APs.
- the expected throughput of each traffic class is easily defined beforehand given how the applications are configured. This is a reasonable assumption in the case of real-time traffic, such as CCTV, VOIP or CBTC. For example, the throughput of a CCTV camera can be accurately estimated by knowing the configured resolution, Frames Per Second (FPS) and bit rate.

D. Handover Prediction

The handover prediction module performs two main actions. First, it downloads logs from all the available WiFi clients via SSH. Second, it parses the logs to extract the features and execute the previously trained model to derive a prediction for the handover. The module exposes a simple function for polling the handover probability for each link.

E. Smart Replication

Before each packet is transmitted, the smart replication module polls the the handover prediction module for the current probability of handover over the packet's assigned link. It consults the policy database for the probability threshold, θ, associated with the flow the packet belongs to. In the case that the probability returned by the handover prediction process exceeds the policy associated with the flow, the VPN client replicates the packet over one of the other links. If more than one link is available, i.e. the train has more than 2 links, it sorts the available links by assigned weight, and then picks the first link where the current handover probability associated with the link is below the threshold required by the application, as defined in the policy database. If no available links fulfill that condition, then it picks the link with the least weight assigned to it.

Since the VPN server does not perform its own handover prediction, is notified of upcoming handovers through the Handover Probability field in the protocol header as shown in Fig 9 available in every packet received from the client. The value in this field is in the range 1-100 and represents a probability of a handover on the link it was received on in the next α seconds. This field is assigned the same handover probability value associated with the link by the client at the moment of transmitting the packet. If the client has not transmitted any packet over a link in over 50 milliseconds, it will transmit an empty packet in order to update the server with the handover probability. The rest of the smart replication process is identical to the client.

V. Experimental Setup

The evaluation environment was created inside a Dell XPS 15 9500 with 32 GBit of ram and an Intel Core i7-10750H 2.60GHz CPU with 6 cores and 12 threads. The VPN client,

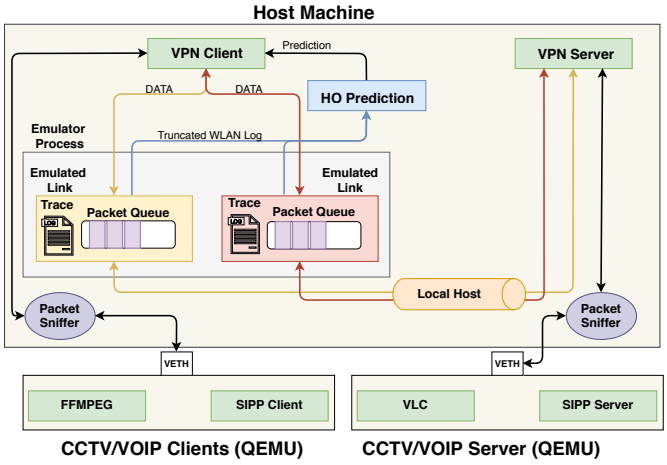


Fig. 10: Emulation architecture

VPN server and the emulator were hosted within the main machine, and communicated using local host. The on-board environment and the network-side environment were hosted within two QEMU [7] virtual machines. The WiFi links were emulated using a custom emulator.

A. Emulator

We designed a custom network emulator that emulates, in real-time, the capacity experienced by a WiFi link during a trip using capacity traces collected from a production system. Additionally, it creates a time-truncated version of the WiFi client log from that trip that is polled by the handover predicito n module, such that it produces a prediction that matches the emulation. An overview of the emulator architecture can be seen in Fig 10.

1) Implementation: The emulator was implemented in C++ as a separate user-space process. Packets originating from the VPN client were re-directed to the emulator using the Linux IPTables [8] with the option TPROXY. The traffic originating from the client was bound to one of two source ports, where each port represents traffic meant for one of the WiFi links.

2) Emulating delay and handovers: Once packets were received by the emulator, they are added to the queue specific to that WiFi link. The queued packet is then delayed sufficiently or dropped to reflect the capacity and handover profile that belonged to that WiFi link. If the packet is not dropped, it will be removed from the queue once its delay duration expires and forwarded to the VPN server. Similarly, traffic coming back from the VPN server would be treated the same way.

The emulator performs a busy wait whenever it has packets in the queue in order to eliminate the need of high resolution timers and the in-accuracy associated with them. The other advantage of this emulator compared to the Linux built-in network emulator NetEm [9] is the ability to switch very efficiently between different rates as the emulation progresses in time without relying on expensive system calls.

3) Emulating RSSI scans: The emulator exposes an API for monitoring a truncated version of the WLAN client log for the trip currently being emulated. The truncation happens based on the current timestamp of the emulation in order to provide the same log that would appear at that point in the real trip. The handover prediction process utilizes the API to obtain the truncated version of the log, and uses it to predict handovers as it would in a real trip.

B. Evaluation configurations

We evaluated the performance of the smart replication system using 5 different values for the θ parameter: 0,0.3,0.5,0.8 and 1. When $\theta = 1$, the system will never replicate packets, and when $\theta = 0$, the system will replicate every packet. With all configurations, the load balancer described in the previous section is used.

C. Application traffic

Since we're concerned with mission-critical real-time traffic in T2G, we evaluated the performance of VOIP calls and CCTV transmission.

1) VOIP setup: We used the SIPP [10] performance testing tool to create multiple parallel VOIP flows that are replaying a PCAP capture from a previous VOIP call we recorded. The performance of the system was evaluated with 5, 10 and 20 parallel VOIP calls. For each of those configurations, 5 separate experiments were performed, on 5 different trips, each lasting for 30 minutes.

2) CCTV setup: We used FFServer [11] to create multiple parallel RTP video streams that replay Big Buck Bunny [12] with three different bit rates: low bit rate (LBR) at 10 mbit/s, medium bit rate (MBR) at 15 mbit/s, and high bit rate (HBR) at 20 mbit/s. For each configuration, 5 separate experiments were performed, on 5 different trips, each lasting for 30 minutes.

D. Evaluation metrics

We collected three metrics in the experiments: loss rate, redundancy rate and latency.

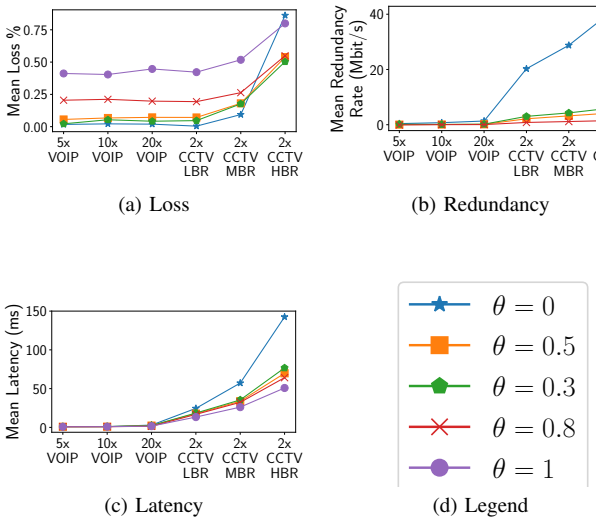
1) Loss: We measure losses by having a packet dump on each of the virtual interfaces connecting the QEMU VMs to the host machine and analyzing the dump at the end of experiment. Since both VOIP and CCTV are RTP traffic, we were able to track the sequence number on the packets on both virtual interfaces to verify if packets with that sequence number were delivered successfully or not.

2) Latency: Similar to loss, we measure latency by using information from the packet dump collected. We calculated the difference in timestamps between packets with the same sequence numbers at both virtual interfaces. Since both packet dumps were running on a single host machine, there were no issues with synchronization and we can simply subtract the timestamps to calculate the latency.

3) Redundancy: For the redundancy, we simply observed how many packets beyond what the originated from the VOIP/CCTV sender were sent by the VPN client. All the extra packets are considered redundancy.

E. Evaluation Results

Fig 11 shows measurements of loss, redundancy and latency for all the schemes with all traffic types. From Fig(a), it can be seen that mean loss for all schemes across all traffic types. The mean loss for $\theta \leq 0.5$ is very similar (<0.1%) when handling 5, 10 and 20 VOIP calls, and the 2x LBR CCTV video, and were all significantly lower than when $\theta \geq 0.8$. In the 2x CCTV MBR CCTV video test case, all schemes



(a) Loss (b) Redundancy

(c) Latency (d) Legend

Fig. 11: (a) mean loss rate vs θ and traffic load. The x-axis shows the traffic load used: 5,10 and 20 parallel VOIP calls, and 2 CCTV flows at 10 Mbit/s each (2x CCTV LBR), 2 CCTV flows at 15 Mbit/s each (2x CCTV MBR) and 2 CCTV flows at 20 Mbit/s each (2x CCTV MBR). (b) mean redundancy rate of replication and smart replication schemes. (c) mean latency rate.

have increased loss due to increased congestion loss. However, for the values of $\theta \leq 0.5$, the increase was higher due to increased number of congestion losses caused by the higher level of redundancy used. This pattern becomes clear in the 2x HBR CCTV test case as the losses in the full replication configuration ($\theta = 0$) exceed the no replication configuration ($\theta = 1$), and the losses of the $\theta = 0.8$ configuration is equal to the $\theta = 0.5$ and $\theta = 0.3$ as the congestion loss and the handover loss balance out. It can be seen that as the application throughput gets closer to the capacity of the link, higher values of θ perform better.

Fig(b) shows mean rate of redundancy for $\theta \leq 0.8$, which are the configurations that use redundancy. In the VOIP test cases, the redundancy from all schemes is fairly small as the VOIP traffic is very light. In the CCTV test cases, the redundancy from the $\theta \geq 0.3$ configurations is less than 10% of the full replication configuration (i.e. $\theta = 0$).

Fig(c) shows the mean latency for all configurations. We can see that the latency overhead caused by replication in all configurations was negligible in all VOIP test cases and in the 2x LBR CCTV case. However, in the remaining cases the latency overhead of the full replication configuration ($\theta = 0$) grows significantly, reaching over 100 ms in the 2x HBR CCTV case. Meanwhile, in the $\theta \geq 0.3$ configuration the overhead is around 5-10 ms.

Overall, it can be seen that in that in all test scenarios, smart replication offers similar or better loss reduction as full replication while providing significantly less redundancy and lower latency. At lower throughput levels, smart replication with $\theta = 0.3$ achieves 91% of the loss reduction provided by simple replication on average while using only 13% of the redundancy and experiencing 20% less latency. As throughput grows relative to capacity, full replication (i.e. $\theta = 0$) causes more congestion losses than it reduces handover losses, making the scheme counterproductive. On the other hand, at $\theta \leq 0.8$ maintains around 34% loss reduction compared to no replication (i.e. $\theta = 1$), but the higher value of θ is able to achieve the same loss reduction with less redundancy, indicating that as throughput grows a higher value of θ performs better as it induces less congestion loss.

VI. Related work

There have been multiple works investigating ways of minimizing the impact of WLAN handovers, both in the high and low mobility environments. We have identified four main categories in the literature. The first category aims to mitigate handovers without attempting to predict them [2] [3]. The second category aims to improve the handovers in WLAN by devising better ways of triggering them [13] [14] [15]. Those approaches are naturally harder to deploy as they require modifying the WLAN clients. The third category are works that combine better handover triggers with cross-layer optimization [16] [17]. The final category, and the closest to our work, are the ones that aim to predict handover, rather than trigger them, and also perform cross-layer optimization [18] [19] [20].

A. Handover mitigation without prediction

In two of the reviewed approaches authors employed constant redundancy to minimize losses from handovers [2] [3].

In [2], authors designed a new multipath protocol RMPTCP based on MPTCP [21] that is better adapted for T2G. RMPTCP replicates all packets by default across all available paths. Congestion control is removed from the subflow level and kept only at the multipath level. TCP re-transmissions are modified such that they only happen if a packet timeout occurs on all the subflows.

In [3], authors propose a redundant mode for MPTCP to be used to transmit data from Supervisory Control and Data Acquisition (SCADA) systems used in trains. The SCADA traffic is latency sensitive and so the authors propose some tuning MPTCP to make it more friendly to real-time traffic. Amongst the optimizations proposed is turning off Nagle [22] algorithm, and effectively eliminating re-transmissions by preventing fast re-transmissions and placing a minimum limit of 64 seconds on return-trip timeout (RTO) retransmissions.

B. Improving handover triggers

Three of the reviewed works dealt with improving handover triggers [13] [14] [15]. In [13] and [14] authors triggered handovers by determining when the RSSI of the currently associated AP will fall below a certain pre-determined threshold. In [13] they used a weighted running average of the RSSI and in [14] they used a neural network with a time-sliding window of the RSSI as input. In [15] authors attempted to address the problem with RSSI variability causing multiple unnecessary handovers, referred to as hysteresis, by relying on link quality rather than RSSI for prediction.

C. Improving handover triggers with cross-layer optimization

Two of the reviewed works combined improving handover triggers with cross-layer optimizations [16] [17].

In [16], authors created a model to predict handovers and use that prediction to lower the bitrate of streamed video

before handovers in order to reduce losses and maintain uninterrupted playback. To predict handovers, they compared the double exponential weighted moving average (DEWMA) of the RSSI of the currently associated AP to one another AP, which and predicted a handover when the two DEWMA crossed.

In [17], authors created a video streaming app that predicts handovers with low granularity, up to 30 seconds, in order to fetch content ahead of time to ensure that no pauses occur during the outage caused by the handover. The predictor collects measurements from the user's device, such as velocity, RSSI, location and direction. It sent this information to a centralized server that hosts a database of historical mobility patterns, which it will then use to try to predict the future location of the device. Once the future location is determined, the server used connectivity map to predict outages based on the location. The server then transmitted the prediction back to the client to decide if it needs to pre-fetch data.

D. Handover prediction and cross-layer optimization

Three of the works reviewed combined handover prediction and cross-layer optimization [20] [18] [19].

In [20] authors designed a reactive approach to predicting handovers during VOIP calls. The approach relies on the rate of WLAN MAC layer retransmissions over a window of time. If a high rate of retransmissions is detected, the proposed system will replicate the traffic over all other available WLAN links. If one of the other available links experiences less MAC layer re-transmissions compared to the current link, the VOIP traffic will be shifted over to that link and the replication will stop.

In [18] authors devised a learning-based approach using a neural network with multiple inputs from smartphone sensors, including RSSI, to predict whether a WiFi handover will occur within 15 seconds. When a handover is predicted, the MPTCP stack would establish a connection over the LTE and use deploy the MPTCP's redundancy scheduler, which will replicate traffic across all links. In [19], authors take the same general approach with handover prediction using a neural network, but they enlarge the scope of inputs used for the prediction to any information that is available beyond sensors, such as link throughput or loss rate.

None of the works reviewed dealt with the challenge of predicting handovers in the underground T2G environment. In particular, the high granularity required of prediction required due to the speed, the more chaotic radio environment due to the tunnels, and the over-the-top deployment inside existing infrastructure.

VII. Conclusion

In this paper, we proposed an over the top approach for minimizing packet loss due to handovers in underground trains equipped with multiple WiFi clients by predicting handovers and replicating packets just before they occur. We devised a learning based handover prediction model based on features extracted exclusively from the WiFi client log. Evaluations of the model on traces from a production system showed that it is able to predict handovers with high accuracy. We integrated our prediction model into a simple adaptive replication scheme and experimentally evaluated variants of the scheme together with no replication and a full replication scheme. Based on the results, we concluded that the smart replication offered a similar or better loss reduction compared to the full replication scheme in all test cases, while utilizing significantly lower redundancy and maintaining lower latency.

References

[1] M. Spiezio, A. Riccio, F. Pezzullo, C. Carcatella, M. Empireo, A. Salemme, A. Montagna, L. D'Alterio, V. Castaldi, and P. Donadio, "Fast radio technologies for uninterrupted train to track side communications." [Online]. Available: https://www.fasttracks.eu/wp-content/uploads/2018/03/deliverable-1-1-fast-tracks-wireless-networks-architecture.pdf

[2] I. Lopez, M. Aguado, and E. Jacob, "End-to-end multipath technology: Enhancing availability and reliability in next-generation packet-switched train signaling systems," vol. 9, no. 1, pp. 28–35.

[3] I. Lopez, M. Aguado, C. Pinedo, and E. Jacob, "SCADA systems in the railway domain: Enhancing reliability through redundant MultipathTCP," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 2305–2310.

[4] I. Lopez, M. Aguado, D. Ugarte, A. Mendiola, and M. Higuero, "Exploiting redundancy and path diversity for railway signalling resiliency," in *2016 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, pp. 432–439.

[5] IEEE 802.11f-2003 - IEEE trial-use recommended practice for multi-vendor access point interoperability via an inter-access point protocol across distribution systems supporting IEEE 802.11 operation. [Online]. Available: https://standards.ieee.org/standard/80211F-2003.html

[6] M. Handley, O. Bonaventure, C. Raiciu, and A. Ford. TCP extensions for multipath operation with multiple addresses. [Online]. Available: https://tools.ietf.org/html/rfc6824

[7] QEMU. [Online]. Available: https://www.qemu.org/

[8] iptables(8) - linux man page. [Online]. Available: https://linux.die.net/man/8/iptables

[9] A. Jurgelionis, J. Laulajainen, M. Hirvonen, and A. I. Wang, "An empirical study of NetEm network emulation functionalities," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–6.

[10] "SIPp/sipp." [Online]. Available: https://github.com/SIPp/sipp

[11] ffserver – FFmpeg. [Online]. Available: https://trac.ffmpeg.org/wiki/ffserver

[12] Big buck bunny. [Online]. Available: https://peach.blender.org/

[13] K. Kumar, P. Angolkar, and R. Ramalingam, "SWiFT: A novel architecture for seamless wireless internet for fast trains," pp. 3011–3015.

[14] M. Feltrin and S. Tomasin, "A machine-learning-based handover prediction for anticipatory techniques in wi-fi networks," in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 341–345.

[15] Tsungnan Lin, Chiapin Wang, and Po-Chiang Lin, "A neural network based context-aware handoff algorithm for multimedia computing," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 2, pp. ii/1129–ii/1132 Vol. 2.

[16] C. Vallati, E. Mingozzi, and C. Benedetto, "Efficient handoff based on link quality prediction for video streaming in urban transport systems: Efficient handoff for video streaming in urban transport systems," vol. 16.

[17] J. Geurts, Z. Li, Y. Liu, and J.-C. Point, "Transparent handover using WiFi network prediction for mobile video streaming," vol. 353, pp. 937–946.

[18] J. Höchst, A. Sterz, A. Frommgen, D. Stohr, R. Steinmetz, and B. Freisleben, "Learning wi-fi connection loss predictions for seamless vertical handovers using multipath TCP," pp. 18–25.

[19] A. Frömmgen, S. Sadasivam, S. Müller, A. Klein, and A. Buchmann, "Poster: Use your senses: A smooth multipath TCP WiFi/mobile handover."

[20] S. Kashihara and Y. Oie, "Handover management based on the number of data frame retransmissions for VoWLAN," vol. 30, no. 17, pp. 3257–3269. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366407000394

[21] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath TCP," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. USENIX Association, pp. 29–29. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228298.2228338

[22] J. Nagle. Congestion control in IP/TCP internetworks. [Online]. Available: https://tools.ietf.org/html/rfc896