THE SKY’S SECRET CHANNELS: UNVEILING ATMOSPHERIC DUCTING IN 6G

Applications of Atmospheric Ducting

1. Point-to-Point Communication Link:

- Atmospheric duct enables beyond line-of-sight (LoS) communications.
- Outperforms traditional relay-based communications in terms of delay, interception risk, detection, and cost efficiency.
- Useful for military and maritime communications between islands and the mainland.

2. Military Communications:

- Atmospheric duct enabled beyond LoS communications have lower transmission delays (a few milliseconds) compared to satellite communications (~500 ms).
- Can operate in a wide frequency range (2 GHz to 20 GHz) with high capacity, outperforming HF radio communications.
- Reduced risk of interception or detection under hostile conditions using low elevation angle beams over long single-hop spans.
- Suitable for tactical operations like rapid strategic deployment, extended radar operation range, and protection from jamming.
- Provides timely, precise, reliable, and secure transmission.

3. Civilian Applications:

- Focus on remote communications between islands and mainland over the sea.
- Cost-effective beyond LoS links for offshore gas or petroleum platforms.
- Advantages over cellular links (infeasible in open seas), submarine optical fiber links (expensive), and satellite links (high operational cost).
- Ideal for continuous transmission needs.
- Successful experimental link: 78 km real-time video transmission between the Great Barrier Reef and the Australian mainland at 10 Mbit/s for more than 80% of the time.

4. Extending Terrestrial Communication Coverage:

- Atmospheric duct enabled beyond LoS communications extend conventional terrestrial communication coverage from ground to air and sea.
- Key component for air-ground-sea integrated networks.
- Enables remote, flexible, and cost-effective connections among UAVs, offshore platforms, and mainland base stations.
- Construction and control of such duct-assisted networks is an ongoing challenge.

Effects of Atmospheric Ducting

1. Channel Modelling:

- Challenge: Analyse the impacts of trapped refraction on wave propagation, different from normal channels in free space.

- Method: Parabolic equation (PE) methods derived from Maxwell equations are used to model atmospheric ducting channels.

- Parameters: PE methods can model multiple parameters including duct thickness, duct strength, frequency, transmitter and receiver heights, and polarisation.

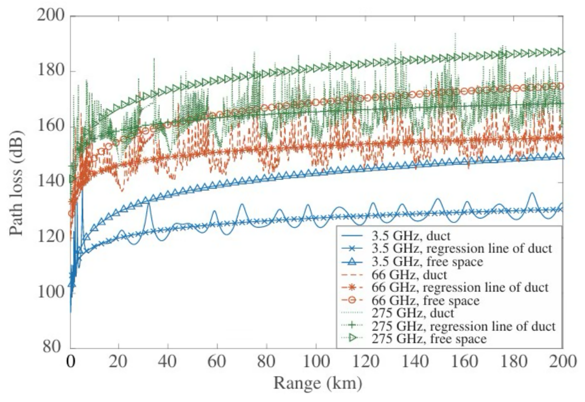


Fig. 3 Path loss in the atmospheric duct and free space at 3.5 GHz, 66 GHz, and 275 GHz

- Path Loss:
    - Obtained using PE methods.
    - Equation: PL=20lg(4 u(xz))+lg(d) 30lg(λ), where u(xz) is the reduced field component, x and z represent range and altitude, d is the range from the transmitter, and λ is the wavelength.
    - Path loss in atmospheric ducting channels is 10-20 dB lower than in free space for sub-6 GHz, mmWave, and THz frequency bands.
    - Lower attenuation in the duct extends communication range, especially at lower frequencies.

- Shadow Fading: Fitted by log-Weibull distribution (Gumbel distribution), unlike the log-normal distribution for normal channels.

- Small-Scale Fading:
    - Fitting Rayleigh distribution within 85%-90% confidence level.
    - Fading correlations in space and polarisation can enhance capacity in atmospheric ducts.

- Modelling: Geometry-based stochastic models describe ducting channels considering beyond LoS, LoS, and reflection paths.

- Comparison:
    - Atmospheric ducting channels have lower path loss, different shadow fading distribution, and similar small-scale fading compared to normal wireless channels.
    - Effective channel estimation algorithms are necessary.

2. Propagation Delay:

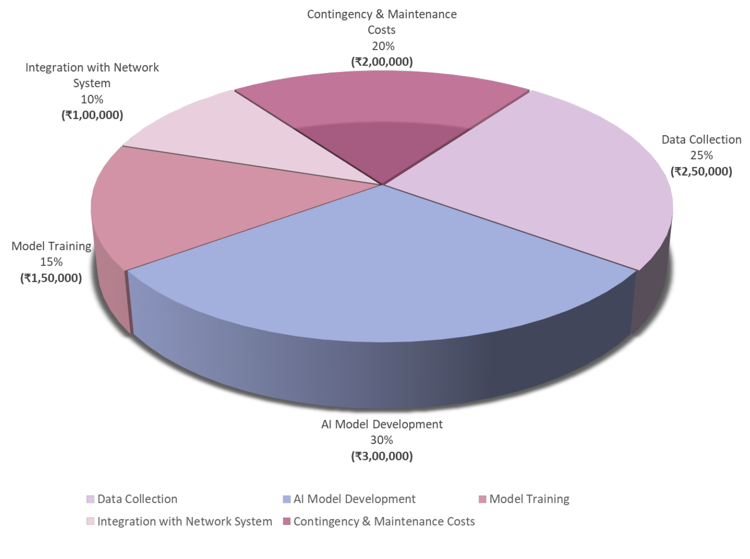
- Delay Characteristics:
    - Large propagation delay due to waves traveling hundreds of kilometres with lower path loss.
    - Different from delay in multi-hop networks; delay in ducting channels is within a point-to-point link.

- Suitability: More suitable for non-real-time services or delay-tolerant applications.
- Signalling Overhead: High signalling overhead for information exchange delay, channel estimation, interference mitigation, and resource management.
- Intelligent Technologies: Introducing semantic communications and distributed learning can enable more effective transmission with smaller data size and delay.
- Connectivity Challenge:
  - Persistent connectivity is fragile due to rigorous formation conditions and dynamic weather or transmission conditions.
  - Estimating the probability of occurrence and duration of the atmospheric duct through meteorological parameters is preferable for better channel state information.

<u>More Detailing on Budget Distribution</u>

1. Data Collection (₹2,00,000)

• Historical Data Purchase or Licensing: ₹50,000
Cost of acquiring historical atmospheric and signal data from meteorological organisations or network providers.
• Real-Time Data Sensors: ₹1,00,000
Purchase and installation of weather sensors and network monitoring tools.
• Satellite Data Subscription: ₹50,000
Subscription to satellite data services for real-time atmospheric monitoring.
• Data Preprocessing (₹50,000)
- Software Tools: ₹30,000
Purchase or subscription to data cleaning and preprocessing tools (e.g., MATLAB, Python libraries).
- Data Cleaning Services: ₹20,000
Cost of data cleaning and preparation services if outsourced.



2. AI Model Development (₹3,00,000)

• Machine Learning Algorithms: ₹1,50,000
Development of machine learning models (including salaries for data scientists and ML engineers).
• Deep Learning Models: ₹1,00,000
Development and training of deep learning models.
• Hybrid Model Integration: ₹50,000
Combining different models for enhanced performance.

3. Model Training (₹1,50,000)

- Computational Resources: ₹70,000

Cost of cloud services or computational hardware for training and validating models (e.g., AWS, Google Cloud).

- Validation Tools and Services: ₹30,000

Tools and external services for validating the models.

- Token Costs: ₹50,000

Assuming a data set occupying 5,00,000 tokens (₹10 for every 1000 tokens as per standardised rate of OpenAI), this budget can process a dataset of roughly 3,50,000-4,00,000 words/inputs.

4. Integration with Network Systems (₹1,00,000)

- Adaptive Routing Implementation: ₹50,000

Integration of AI predictions with network routing protocols and algorithms.

- Resource Allocation Optimisation: ₹25,000

Development of systems to optimise resource allocation based on AI predictions.

- Alerts and Notifications System: ₹25,000

5. Contingency and Maintenance Costs (₹2,00,000)

- Contingency Fund: ₹1,00,000
    - Real-Time Monitoring Software: ₹30,000

Development or purchase of software for real-time prediction and monitoring.

    - Anomaly Detection Systems: ₹20,000

Implementation of systems to detect signal anomalies, reserved for unforeseen expenses or additional costs.

    - Feedback Loop Implementation: ₹50,000

Setting up systems for continuous learning and model updates.

- Model Maintenance and Updates: ₹50,000

Regular updates and maintenance of AI models.

- Miscellaneous Expenses: ₹50,000

Considerations

- Customisation: Specific requirements and customisation may alter the budget allocation.
- Scalability: Ensure the solution is scalable for future enhancements and increased data volumes.
- Vendor Selection: Choosing the right vendors for data, cloud services, and software tools can impact costs.
- Skillset: Hiring skilled professionals or partnering with expert agencies can affect development and integration expenses.

<u>Detailing of the AI-Prototype</u>

Designing an AI prototype that combines Random Forest, LSTM, and CNN-LSTM models to aid in implementing atmospheric ducting involves several steps, including data preprocessing, model architecture design, training, and fine-tuning. Here's a step-by-step outline:

Step 1: Data Collection and Preprocessing

Gather relevant data related to atmospheric conditions, such as temperature, humidity, pressure, and other environmental factors. This data should be collected over time to capture temporal patterns.

1. **Data Cleaning**: Remove any missing or erroneous data points.
2. **Normalisation**: Scale the data to ensure it fits within a specific range, typically [0, 1].
3. **Feature Engineering**: Create additional features that may help in predicting atmospheric ducting, such as temporal features (time of day, season) or derived features (temperature gradients).

Step 2: Model Design

1. Random Forest Model

A Random Forest can be used to capture non-linear relationships and feature importance.

```
from sklearn.ensemble import RandomForestRegressor

# Initialize Random Forest
rf_model = RandomForestRegressor(n_estimators=100,
max_depth=10, random_state=42)

# Fit the model
rf_model.fit(X_train, y_train)

# Predict
rf_predictions = rf_model.predict(X_test)
```

2. LSTM Model

LSTM networks are suitable for capturing long-term dependencies in time-series data.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout

# Initialize LSTM model
lstm_model = Sequential([
    LSTM(50, activation='relu', input_shape=(n_timesteps,
n_features)),
    Dropout(0.2),
    Dense(1)
])

# Compile the model
lstm_model.compile(optimizer='adam', loss='mse')

# Fit the model
lstm_model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.2)
```

3. CNN-LSTM Hybrid Model
This model combines convolutional layers with LSTM layers to capture both spatial and temporal features.

```
from tensorflow.keras.layers import Conv1D, MaxPooling1D,
Flatten

# Initialize CNN-LSTM model
cnn_lstm_model = Sequential([
    Conv1D(filters=64, kernel_size=2, activation='relu',
input_shape=(n_timesteps, n_features)),
    MaxPooling1D(pool_size=2),
    Flatten(),
    LSTM(50, activation='relu'),
    Dropout(0.2),
    Dense(1)
])

# Compile the model
cnn_lstm_model.compile(optimizer='adam', loss='mse')

# Fit the model
cnn_lstm_model.fit(X_train, y_train, epochs=50,
batch_size=32, validation_split=0.2)
```

Step 3: Fine-Tuning Procedures

1. Hyperparameter Tuning
Use techniques such as Grid Search or Random Search to find the optimal hyper-parameters for each model.

```
from sklearn.model_selection import GridSearchCV
```

```
# Example for Random Forest
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20],
    'min_samples_split': [2, 5]
}

grid_search = GridSearchCV(estimator=rf_model,
param_grid=param_grid, cv=3, n_jobs=-1)
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
rf_model = grid_search.best_estimator_
```

2. Cross-Validation
Implement k-fold cross-validation to ensure the model's robustness and to avoid overfitting.

```
from sklearn.model_selection import cross_val_score

# Cross-validation for Random Forest
cv_scores = cross_val_score(rf_model, X_train, y_train, cv=5)
print(f"Average CV Score: {np.mean(cv_scores)}”)
```

3. Ensemble Learning
Combine the predictions from different models to improve accuracy.

```
from sklearn.ensemble import VotingRegressor

# Create an ensemble of models
ensemble_model = VotingRegressor([('rf', rf_model), ('lstm',
lstm_model), ('cnn_lstm', cnn_lstm_model)])

# Fit the ensemble model
ensemble_model.fit(X_train, y_train)

# Predict
ensemble_predictions = ensemble_model.predict(X_test)
```

4. Model Evaluation
Evaluate each model using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared.

```
from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score
```

```
# Evaluate Random Forest
mse_rf = mean_squared_error(y_test, rf_predictions)
mae_rf = mean_absolute_error(y_test, rf_predictions)
r2_rf = r2_score(y_test, rf_predictions)

# Evaluate LSTM
lstm_predictions = lstm_model.predict(X_test)
mse_lstm = mean_squared_error(y_test, lstm_predictions)
mae_lstm = mean_absolute_error(y_test, lstm_predictions)
r2_lstm = r2_score(y_test, lstm_predictions)

# Evaluate CNN-LSTM
cnn_lstm_predictions = cnn_lstm_model.predict(X_test)
mse_cnn_lstm = mean_squared_error(y_test,
cnn_lstm_predictions)
mae_cnn_lstm = mean_absolute_error(y_test,
cnn_lstm_predictions)
r2_cnn_lstm = r2_score(y_test, cnn_lstm_predictions)

# Evaluate Ensemble
mse_ensemble = mean_squared_error(y_test,
ensemble_predictions)
mae_ensemble = mean_absolute_error(y_test,
ensemble_predictions)
r2_ensemble = r2_score(y_test, ensemble_predictions)

print(f"RF MSE: {mse_rf}, LSTM MSE: {mse_lstm}, CNN-LSTM MSE:
{mse_cnn_lstm}, Ensemble MSE: {mse_ensemble}")
```

Conclusion

The designed AI prototype leverages the strengths of Random Forest, LSTM, and CNN-LSTM models to effectively predict atmospheric ducting conditions. By combining these models and using advanced fine-tuning techniques, the prototype can provide accurate and robust predictions, aiding in the implementation of atmospheric ducting. This approach ensures that the model captures both spatial and temporal patterns, which are crucial for understanding atmospheric phenomena.