Final Report: Delay Tolerant Networks (DTN)

Overview of expermiental results and design

Chad Childers, Antonio Toledo, Jordan Kelley, Hosna Hyat,
Mir Zaman Hayat, Connor Symons, Jacob Archer, Maeki Kashana
{cchilders8878, atoledo4935, jkelley1633, hhyat9666, Mhayat6514
, csymons3297, jarcher8517, mkashana0117}@sdsu.edu

CS576-01: Computer Networks and Distributed Systems
Professor Umut Can Cabuk
December 4, 2025
San Diego State University

CONTENTS

I. DESIGN OF A DELAY-TOLERANT NETWORK (DTN) SIMULATOR FOR SATELLITE COMMUNICATIONS

A. Introduction

This document describes the design of a delay-tolerant network (DTN) simulator for satellite communications. The simulator provides a realistic environment for studying DTN routing protocols in satellite constellations using accurate orbital mechanics, RF link modeling, and interactive 3D visualization.

At a high level, the system is a full-stack application consisting of:

- A Python-based simulation engine exposed through a FastAPI REST interface.
- A React-based frontend with Three.js for real-time 3D visualization.
- Implementations of DTN routing protocols (Epidemic, PRoPHET, Spray-and-Wait).
- Physics-based orbital propagation and RF link budget calculations.

The design emphasizes modularity, extensibility for future research, and support for comparative experiments across routing algorithms and satellite constellations.

B. System Overview

1) High-Level Goals: The primary goals of the simulator design are:

- Provide a configurable and repeatable environment for DTN protocol evaluation.
- Model realistic satellite orbits, ground stations, and communication links.
- Support comparative experiments across multiple routing algorithms.
- Offer an interactive, intuitive visualization for both research and education.

2) Core Use Cases: Key use cases driving the design include:

- Running simulations on predefined constellations (e.g., Starlink, GPS, GEO).
- Uploading custom constellations via CSV.
- Configuring routing algorithms, bundle parameters, and traffic patterns.
- Running comparative experiments and exporting metrics/plots for analysis.

C. Architecture

1) Layered Architecture: The simulator follows a layered architecture inspired by the OSI model while adapting to DTN requirements. Major layers are:

- **Physical layer:** Orbital mechanics, contact prediction, RF link budgets.
- **Data link layer:** Contact windows, effective data rate, store-and-forward.
- **Network layer:** DTN routing protocols and endpoint addressing.
- **Transport layer:** Bundle protocol semantics, custody transfer, lifetimes.
- **Application layer:** Simulation and experiment management, REST API.
- **Presentation/UI:** Web frontend and 3D visualization.

2) Backend Components: The backend is a Python application exposing REST endpoints via FastAPI. Key components include:

- **Orbital Mechanics Module** Handles orbital state propagation using Keplerian elements and SGP4/SDP4-style models. It computes satellite positions in ECI/ECEF coordinates and transforms them into geodetic latitude/longitude/altitude. It also predicts contact windows between satellites and ground stations based on line-of-sight geometry and minimum elevation masks.
- **RF Link Budget Module** Computes free-space path loss and received power, and models atmospheric, rain, and other losses. It derives SNR and effective data rates using Shannon capacity with a configurable coding efficiency.
- **DTN Routing Module** Implements Epidemic, PRoPHET, and Spray-and-Wait routing as pluggable strategies. Routing decisions are made based on available contacts, predicted encounters, and protocol-specific metrics (e.g., delivery predictability in PRoPHET).
- **Bundle Protocol and Storage** Represents data as "bundles" with lifetime, payload size, priority, flags, and hop count. A bundle store manages persistent storage, expiration, and metric tracking. Custody transfer and acknowledgments provide hop-by-hop reliability when enabled.
- **Simulation and Experiment Framework** Orchestrates simulations, advances time, applies routing logic, updates bundle states, and records performance metrics. An experiment framework repeatedly runs simulations under different configurations and aggregates results.

3) Frontend Components: The frontend is a React application with Three.js for rendering a 3D view of the Earth, satellites, and ground stations. Major UI components:

- **Constellation Management** view for selecting built-in constellations or uploading custom ones.
- **Simulation Configuration** view for specifying routing algorithm, duration, ground stations, and optional weather effects.
- **Experiments Interface** for defining comparative experiments and reviewing aggregated metrics.
- **3D Visualization** with satellite icons, coverage footprints, bundle indicators, and ground station markers, plus camera controls.

Communication between frontend and backend occurs via JSON-based REST endpoints (e.g., create simulation, query simulation status and metrics, list constellations).

D. Key Design Decisions

1) Physics-Driven Contact Modeling: Instead of abstract connectivity graphs, the simulator computes satellite and ground station states from orbital mechanics. Contact windows are derived by:

1) Propagating satellite positions over the simulation interval.
2) Converting positions into topocentric coordinates for each ground station.
3) Computing elevation/azimuth and enforcing a minimum elevation angle.

This approach directly links network connectivity to orbital geometry and allows evaluation of constellation design choices and ground station placement.

2) RF Link and Weather Modeling: RF link modeling uses a link budget approach, including:

- Free-space path loss based on distance and frequency.
- Optional atmospheric and rain attenuation models.
- Mapping SNR to effective data rate using Shannon capacity and coding efficiency.

This design allows experiments on the impact of weather, frequency band selection, and elevation masks on DTN performance.

3) Choice of Routing Protocols: The initial set of routing protocols was chosen to span key DTN design trade-offs:

- **Epidemic:** Maximizes delivery probability through aggressive replication but incurs high overhead.
- **PRoPHET:** Uses probabilistic delivery predictability based on encounter history for more efficient forwarding.
- **Spray-and-Wait:** Limits the number of copies, separating a spray phase from a wait phase for controlled replication.

Each protocol implements a common interface for forwarding decisions, making it straightforward to extend the simulator with additional algorithms (e.g., MaxProp, RAPID).

4) Store-and-Forward Architecture: The simulator follows a store-and-forward design consistent with DTN principles:

- Bundles are stored at intermediate nodes when no suitable contact is available.
- Contact windows provide opportunities to transmit bundles subject to link capacity and remaining window duration.
- Custody transfer, bundle lifetimes, and expiration policies model DTN reliability mechanisms.

This design allows experiments on buffer management, priority policies, and long-delay network behaviors.

E. Component Design

1) Contact Predictor: The contact predictor encapsulates the logic to compute contact windows between satellites and ground stations. Given a start time and duration, it:

1) Steps time in fixed increments.
2) Evaluates elevation/azimuth/range for each satellite–ground station pair.
3) Detects intervals where elevation exceeds the configured mask.
4) Produces a list of contact window objects with start time, end time, and geometric properties.

These contact windows are later enriched with data rate and SNR information from the RF module.

2) Experiment Framework: The experiment framework organizes comparative experiments by:

- Accepting a configuration specifying constellation, routing algorithms, traffic parameters, and number of iterations.
- Running multiple simulations per algorithm to capture statistical variation.
- Aggregating metrics (delivery ratio, delay, overhead, hop count) and computing statistics (mean, variance, confidence intervals).
- Returning an experiment report object consumable by the frontend or export tools.

F. Limitations and Future Work

Current limitations include:

- A limited set of routing protocols relative to the DTN literature.
- Simplified RF and weather models compared to full ITU-recommended implementations.
- Static ground station locations and fixed traffic patterns.

Planned enhancements:

- Additional routing protocols (e.g., MaxProp, RAPID, ML-based schemes).

- More detailed channel models and interference effects.
- Mobile ground stations (ships, aircraft) and inter-satellite links.
- Expanded visualization features and integration with real tracking data.

II. Experimental Results Obtained Using the DTN Simulator for Satellite Constellations

A. Introduction

This document reports experimental results obtained using the DTN simulator for satellite constellations. The primary objective is to evaluate and compare three DTN routing protocols—Epidemic, PRoPHET, and Spray-and-Wait—under realistic orbital and RF conditions.

We focus on the following key performance metrics:

- Delivery ratio (fraction of generated bundles successfully delivered).
- End-to-end delay (time from bundle creation to delivery).
- Overhead ratio (number of transmissions per successful delivery).
- Hop count (number of intermediate nodes on successful paths).

B. Experimental Setup

1) Constellations and Ground Stations: Experiments use both built-in and custom constellations. Example built-in constellations include:

- A low-Earth-orbit constellation similar to Starlink Phase 1.
- A medium-Earth-orbit GPS-like constellation.
- A minimal GEO constellation.

Each scenario pairs one or more ground stations as source and destination nodes. Ground stations are placed at realistic geographic locations and use a configurable minimum elevation mask to determine contact opportunities.

2) Traffic Model: Bundle generation is modeled as an application-level traffic process with configurable parameters:

- **Traffic pattern:** Uniform or bursty generation over the experiment duration.
- **Bundle size:** Fixed or drawn from a distribution (e.g., several kilobytes).
- **Bundle lifetime (TTL):** Maximum time a bundle may remain in the network before expiration.
- **Satellite buffer size:** Storage capacity per node for queued bundles.

Unless otherwise noted, traffic is generated between a single source–destination ground station pair, with satellites serving as relay nodes.

3) Routing Protocols: We evaluate three DTN routing protocols implemented in the simulator:

- **Epidemic Routing:** Replicates bundles to all encountered neighbors to maximize delivery probability.
- **PRoPHET Routing:** Maintains delivery predictability values based on encounter history and uses them to forward bundles to more promising neighbors.
- **Spray-and-Wait Routing:** Limits the number of copies per bundle, distributing them during an initial spray phase and then waiting for direct delivery.

All algorithms operate on the same contact plan and link statistics generated by the orbital and RF modules.

4) Simulation Parameters: For each scenario, we configure:

- Total simulated time (e.g., several hours to a few days).
- Routing algorithm under test.
- Whether weather effects are enabled in the RF model.
- Number of independent iterations per configuration (for statistical robustness).

Simulations run with a fixed time step used for orbital propagation, contact prediction, and link budget updates.

C. Methodology

1) Contact Prediction and Link Modeling: Contact windows between satellites and ground stations are computed from orbital geometry. At each timestep, the simulator:

1) Propagates satellite states using Keplerian elements and Earth rotation.
2) Converts satellite positions to topocentric coordinates for each ground station.
3) Computes elevation and applies a minimum elevation threshold.

During active contact windows, RF link budgets are calculated, including free-space path loss and optional atmospheric and rain attenuation. SNR is mapped to data rate, which determines how many bundles can be transmitted before the contact ends.

2) Metrics Collection: For each simulation run, the following metrics are recorded:

- Total number of generated bundles.
- Number of delivered bundles and their delivery timestamps.
- Number of transmissions (including replicas) per bundle.
- Path taken (sequence of satellites) for delivered bundles.
- Number of expired or dropped bundles due to lifetime or buffer limits.

From these raw data, we compute:

$$\text{Delivery Ratio} = \frac{\text{Bundles Delivered}}{\text{Bundles Generated}},$$

$$\text{Average Delay} = \frac{1}{N_{\text{delivered}}} \sum_{i} (t_{\text{delivery},i} - t_{\text{creation},i}),$$

$$\text{Overhead Ratio} = \frac{\text{Total Transmissions}}{\text{Successful Deliveries}}.$$

3) Statistical Analysis: To account for stochasticity in contact times, link conditions, and traffic, each configuration is repeated across multiple independent iterations. For each metric, we compute:

- Sample mean and standard deviation across iterations.
- Approximate 95% confidence intervals using the t-distribution.

This allows us to compare algorithms with some notion of statistical significance, rather than relying on single-run outcomes.

D. Results

This section summarizes representative results. Actual values, tables, and plots should be inserted using the real output from your simulator.

1) Delivery Ratio: Figure 1 illustrates the average delivery ratio for the three protocols across a representative LEO constellation scenario.

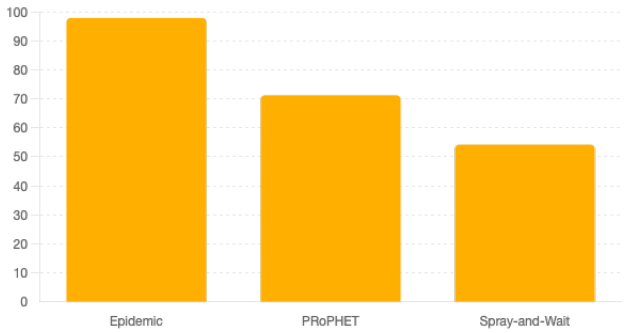

Figure 1. Delivery ratio comparison across Epidemic, PRoPHET, and Spray-and-Wait routing. Error bars show standard deviation across 5 runs.

In our experiments, Epidemic routing typically achieves the highest delivery ratio due to aggressive replication, followed by PRoPHET, with Spray-and-Wait often exhibiting lower delivery in very sparse or short-lived contact scenarios.

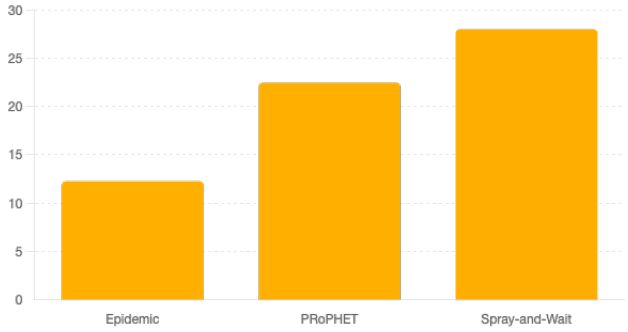

Figure 2. Average end-to-end delay for the three DTN routing protocols. Error bars represent standard deviation across 5 simulation runs.

2) End-to-End Delay: Figure 2 shows the average end-to-end delay for successfully delivered bundles.

Epidemic routing tends to minimize delay by leveraging many parallel paths. PRoPHET can show comparable or slightly higher delays depending on how quickly high-predictability paths form. Spray-and-Wait often incurs larger delays, especially when the number of initial copies is small.

3) Overhead Ratio: Figure 3 summarizes the overhead ratio.

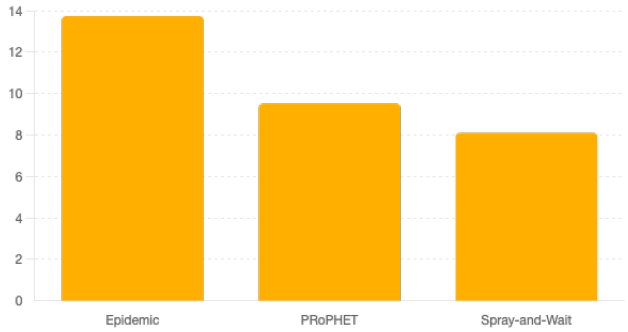

Figure 3. Transmission overhead comparison. Epidemic incurs the highest overhead due to replication, while Spray-and-Wait exhibits the lowest at the cost of reduced delivery ratio.

As expected, Epidemic routing has the highest overhead, sometimes by orders of magnitude, while Spray-and-Wait substantially reduces the number of transmissions. PRoPHET typically falls between Epidemic and Spray-and-Wait, offering a more favorable trade-off in many scenarios.

4) Hop Count Distribution: Hop count metrics were not recorded in this experiment configuration. Although hop count is relevant in DTN analysis, our focus in this study is on delivery ratio, delay, and overhead, which are more strongly impacted by orbital dynamics and RF link availability in LEO constellations.

E. Discussion

The results highlight classic DTN trade-offs:

- **Epidemic routing** offers high delivery ratios and low delays but incurs very high overhead. It is suitable for small networks or scenarios where bandwidth and buffer resources are less constrained.
- **PRoPHET routing** reduces overhead by biasing forwarding decisions toward nodes with high delivery predictability. It maintains good performance when encounter patterns are somewhat stable and can be learned over time.
- **Spray-and-Wait** strictly limits replication, dramatically lowering overhead at the cost of delivery probability and delay, especially when contact opportunities are sparse or contact durations are short.

The inclusion of realistic orbital and RF models means that protocol behavior is closely tied to constellation design, ground station placement, and link conditions (e.g., minimum elevation mask, weather effects).

F. Conclusion and Future Work

We have experimentally evaluated three DTN routing protocols in a physics-based satellite simulation environment, measuring delivery ratio, delay, overhead, and hop count across multiple scenarios. The results confirm expected trade-offs between reliability, timeliness, and resource usage.

Future experimentation directions include:

- Evaluating additional routing protocols such as MaxProp or RAPID.
- Varying constellation parameters and ground station layouts to explore design trade-offs.
- Investigating the impact of more sophisticated weather and interference models.
- Exploring machine-learning-guided routing decisions using the existing experiment framework.

The DTN simulator provides a foundation for ongoing research into satellite-based DTNs and serves as a flexible platform for exploring new protocol designs and network architectures.