



NAMED DATA
NETWORKING

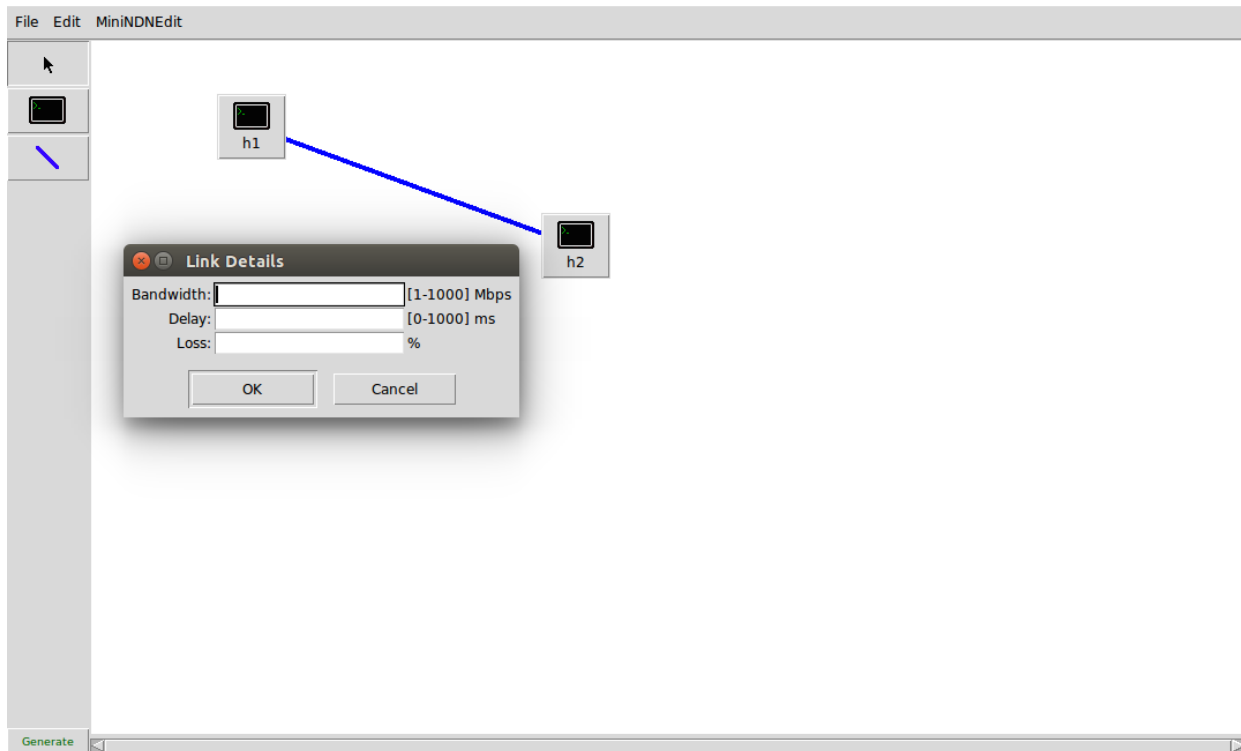
MiniNDN GUI with NDN-Play *[Experimental]*

TUTORIAL:

NDN EVALUATION TOOLS: NDNSIM AND MINI-NDN

Debugging with MiniNDN

- MiniNDN is good for writing test scripts
- But CLI not designed for debugging
- Existing GUI is primitive, needs X11



NDN-Play: <https://play.ndn.today>

- Started as an educational tool
- NDN simulator completely in the browser
- Runs simple NDN experiments

The screenshot displays the NDN-Play web interface. On the left, there is a sidebar with the title "NDN-Play" and "Global Operations". It includes a "Compute Routes" button and several input fields for configuration: "Default Latency (ms)" set to 10, "Default Loss (%)" set to 0, "Content Store Size" set to 500, and "Latency Slowdown Multiplier" set to 10. There is also a checkbox for "Enable Packet Capture (all nodes)" which is currently unchecked. Below these are three buttons: "Run" (green), "Load" (red), and "Generate" (blue). At the bottom of the sidebar is a "MiniNDN Config:" section with a text area and a "Load" button.

The main area of the interface shows a network topology diagram with nodes labeled A, B, C, D, and E. Node A is connected to nodes B, C, D, and E. Node B is connected to node A. Node C is connected to node A. Node D is connected to nodes A and E. Node E is connected to nodes A and D. There is an "Edit" button in the top left of the topology area. Below the diagram, there are tabs for "Console", "TLV Visualizer", and "Traffic Replay". The console shows the following output:

```
Compiled TLV types
Computing routes
```

NDN-Play Features: Visual Topology Editor

- Add / remove nodes and links
- Set link latency / loss rates
- Import / Export MiniNDN config
- Visualize traffic hotspots

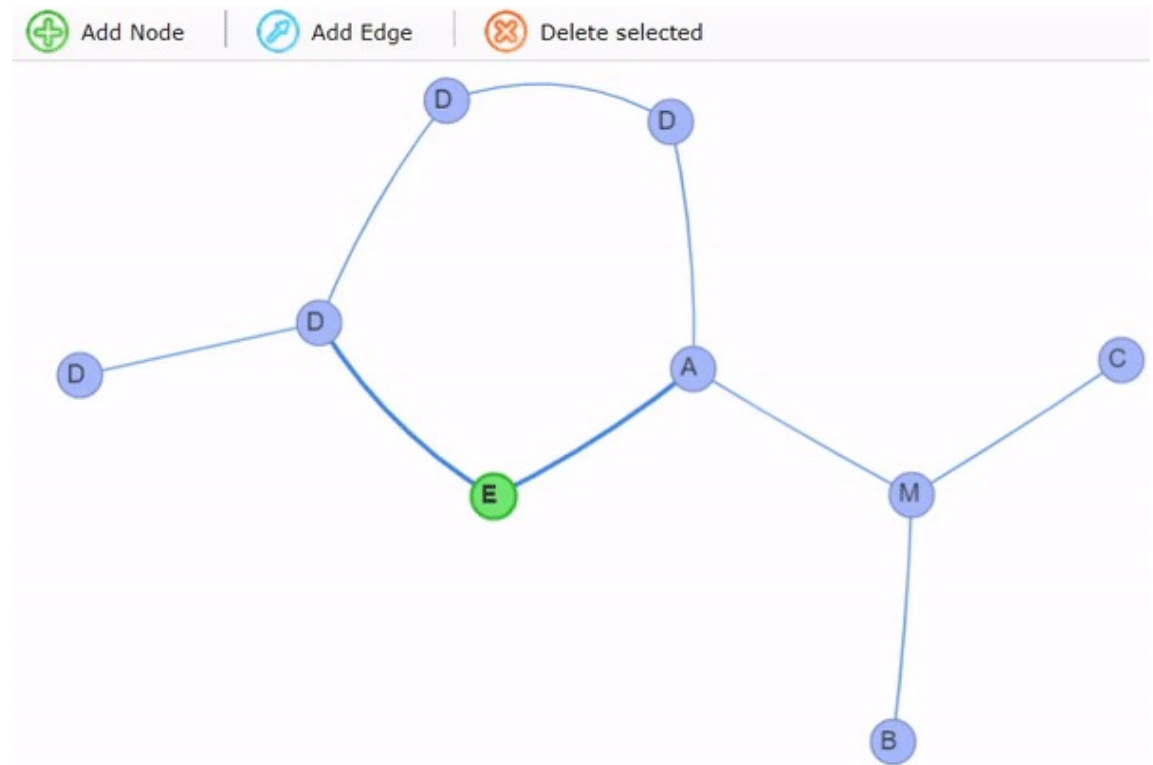
Link A — M

Latency (ms)

Negative for default latency

Loss (%)

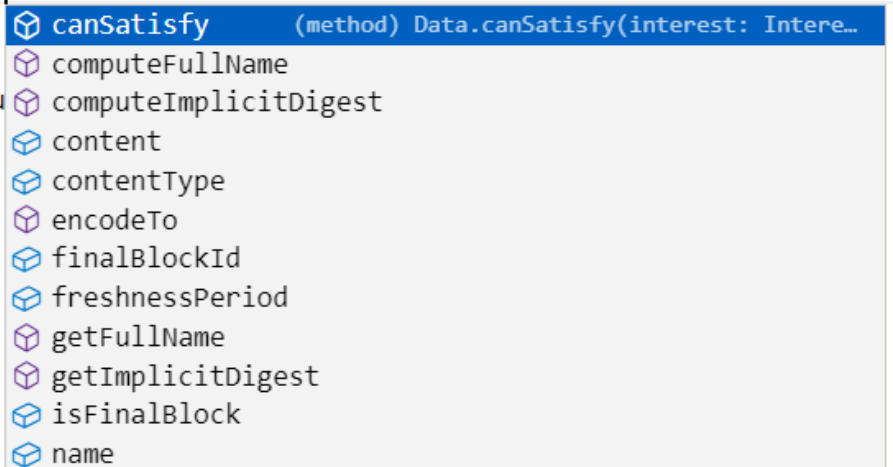
Negative for default loss



NDN-Play Features: TS Editor with Intellisense

- Write NDN-ts code for nodes
- Autocompletion and error-check
- Provides a simplistic forwarder
 - Run code in-browser

```
1  const { Data, Interest } = ndn.packet;
2  const { fromUtf8, toUtf8 } = ndn.util;
3
4  const exec = async () => {
5      const endpoint = node.nfw.getEndpoint({ secure: false });
6      const interest = new Interest('/ndn/producer/test');
7      const data = await endpoint.consume(interest);
8      alert(`${node.label} received "${fromUtf8(data.content)}"`);
9      data.
10 };
11
12 setTimeout(
13
```



The image shows a code editor with a dropdown menu (Intellisense) open over the text 'data.' on line 9. The dropdown lists various properties and methods of the Data object, including 'canSatisfy', 'computeFullName', 'computeImplicitDigest', 'content', 'contentType', 'encodeTo', 'finalBlockId', 'freshnessPeriod', 'getFullName', 'getImplicitDigest', 'isFinalBlock', and 'name'. The 'canSatisfy' method is highlighted in blue.

NDN-Play Features: Packet Capture

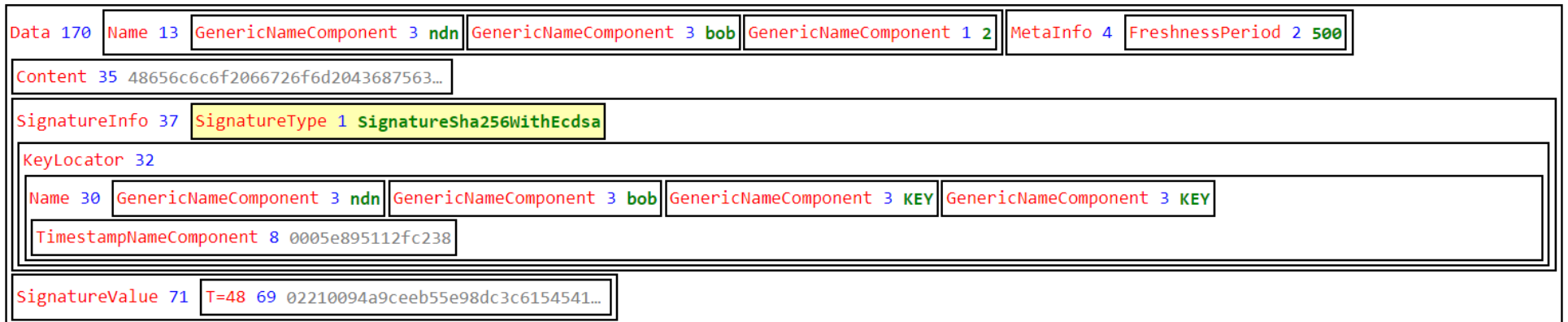
- Capture selectively at nodes or everywhere
- Quickly study data flow



Timestamp	Type	Length	Name
150048 ms	Interest [Consumer → Forwarder]	31	/ndn/producer/test
150151 ms	Interest [Forwarder → Producer]	31	/ndn/producer/test
150259 ms	Data [Producer → Forwarder]	64	/ndn/producer/test
150262 ms	Data [Forwarder → Consumer]	64	/ndn/producer/test
256254 ms	Interest [Consumer → Forwarder]	31	/ndn/producer/test
256356 ms	Interest [Forwarder → Producer]	31	/ndn/producer/test
256460 ms	Data [Producer → Forwarder]	64	/ndn/producer/test
256461 ms	Data [Forwarder → Consumer]	64	/ndn/producer/test

NDN-Play Features: TLV Visualizer

- Peek inside packets from capture
- Decode common types such as SignatureType
- Support custom TLV types for application data



NDN-Play: Missing

- Can only run JavaScript
 - Cannot use ndn-cxx / python-ndn
 - Result: cannot use with real code (in most cases)
- Constrained resources by browser
- Uses a dumb forwarder

Integrating NDN-Play with MiniNDN

- Connect NDN-Play to MiniNDN over WebSocket
- Pull NFD and node stats from server
- Capture and serve packet traces with Tshark
- Limited topology manipulation (experimental)
- Provide a **terminal emulator** on each node for debugging

Terminal Emulator

- Makes it easy to use MiniNDN for real-time debugging
- Manually run code within each node's network namespace
 - Can be C++ / python / scripts
- Make quick file edits
 - E.g. node-specific configuration
 - PTY allows running Vim / nano
 - Not possible in MiniNDN CLI

Demo

- `ssh -L8765:localhost:8765 user@server`
- `docker run -p 8765:8765 --rm -it --privileged ghcr.io/pulsejet/mini-ndn:master`

Thanks

- Try out NDN-Play and the MiniNDN integration
- Contribute
 - <https://github.com/pulsejet/ndn-play>
 - <https://github.com/pulsejet/mini-ndn>
- Questions?