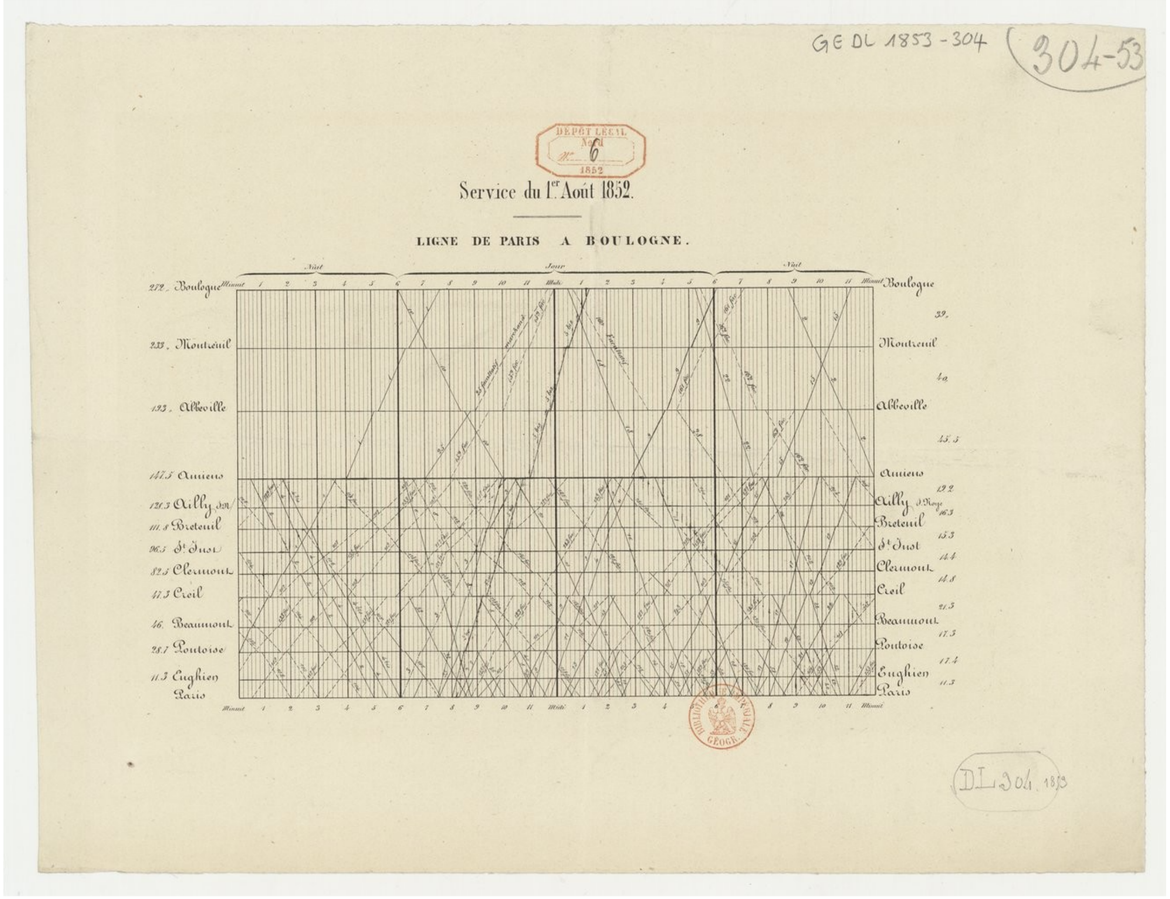
Real-Time Service Analysis for the London Underground: Trackernet Stringlines

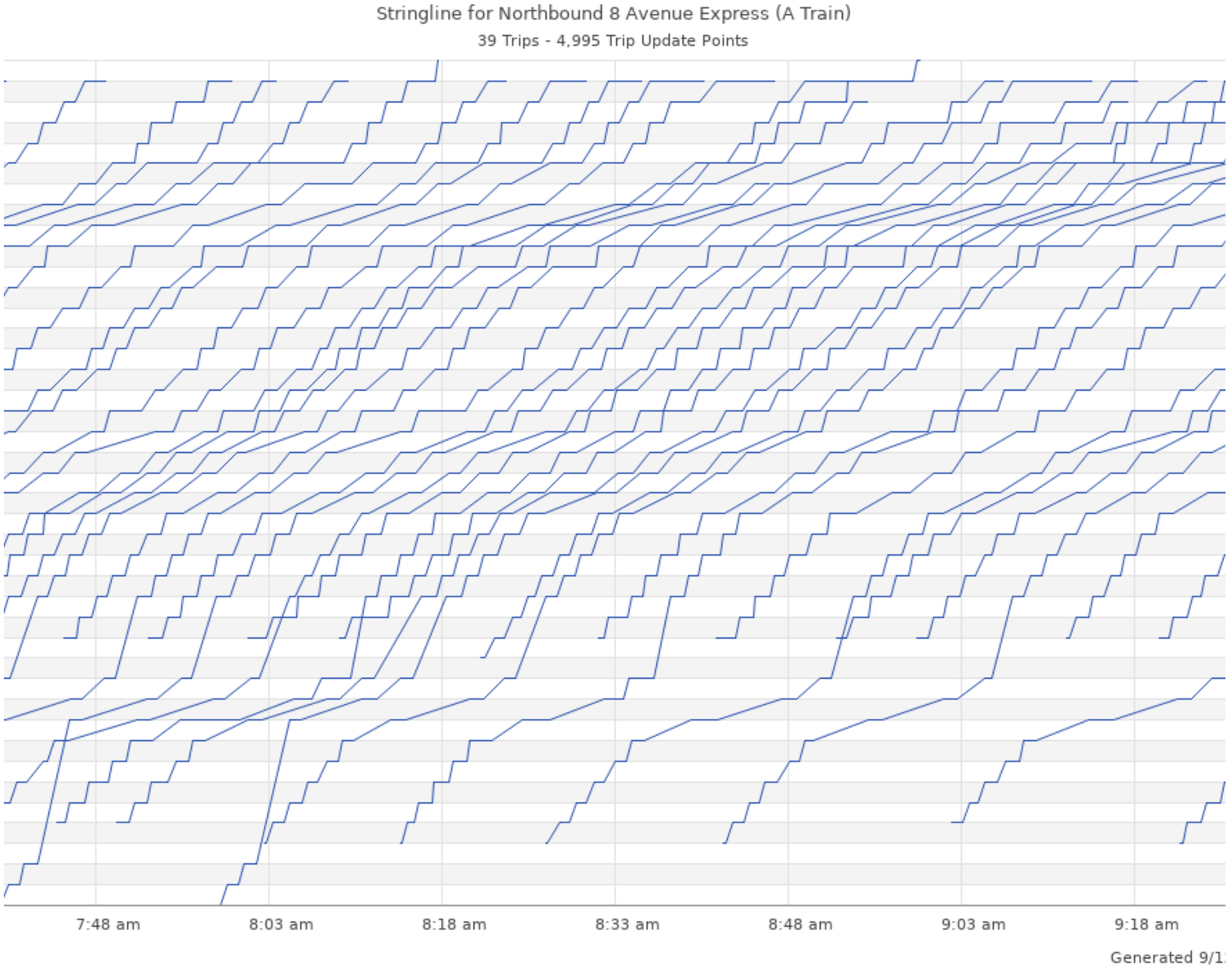
Kurt Raschke, TransportationCamp New England, September 2025

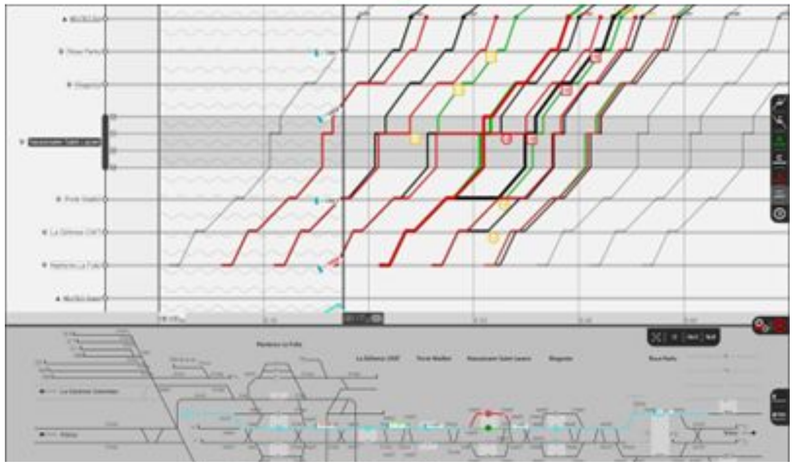
What’s a stringline diagram?

- Developed by Charles Ibry in mid-1800s France
- X-Y plot with time on one axis, stations on the other
- Easily illuminates gaps, bunching, conflicts, etc.
- Considered a primary analytical tool for service management



Source gallica.bnf.fr / Bibliothèque nationale de France





Select Line: | Blue Line | Green Line | Yellow Line

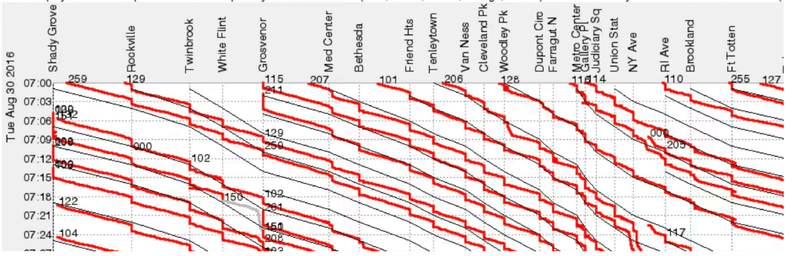
Select Date: Aug 30 2016 **Select Times (0-24HRS): From** 07:00 **To** 09:00 *(2 hour range provides best display)*

Select Track: 1 2 **Identify Stations on Display by:** RTU Station Name Chain Marker
Show each train: ID length in cars destination code

When done, click: Generate Graph

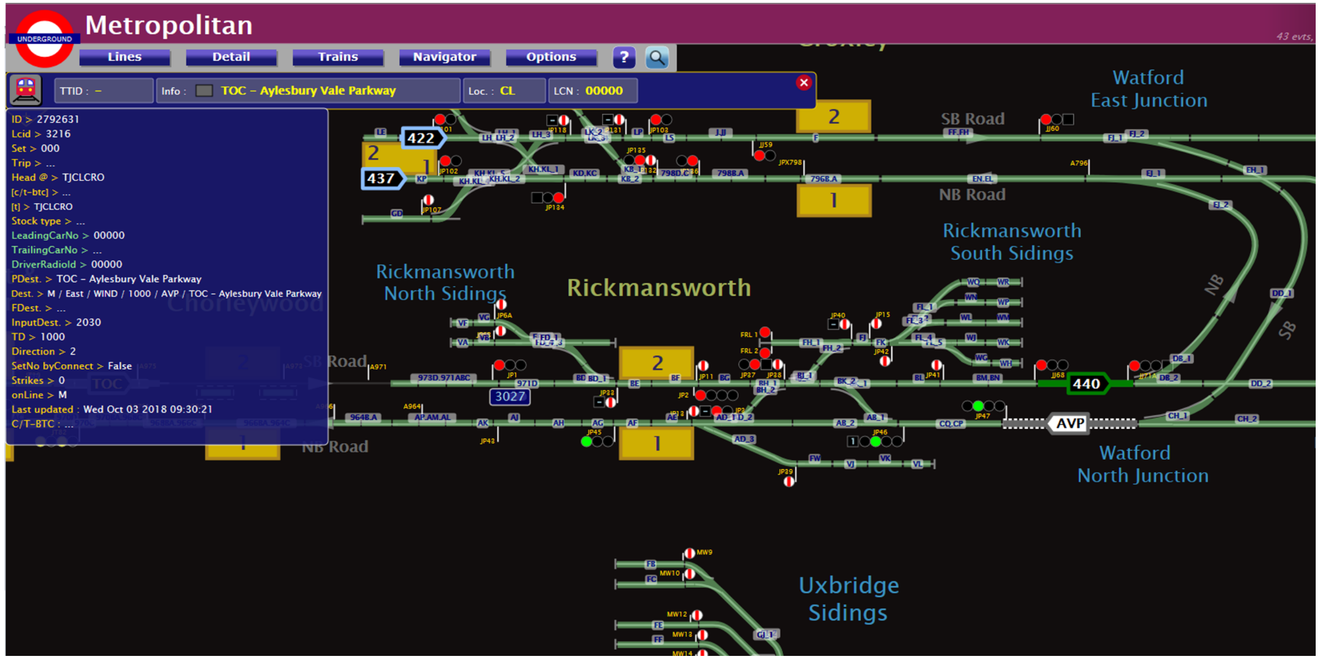
Notes: Thin black diagonal lines on display show planned schedule. Colored lines indicate actual train schedule performance. Numbers at ends of colored lines a
500 series = Green Line, 700 series = Transport or gap trains, 900 series = Orange Line, Gray lines indicate trains with non-specific destination codes, P series
schedule display lines slope down from right to left. Light gray chart background graph lines show relative distance between stations.

Chart displays schedule & performance for the primary train line selected (Blue, Green, Yellow, Orange, or Red) as well as for any other line which shares track (



What is Trackernet?

- Internal London Underground tool for train tracking, using data from train control systems (legacy and CBTC) plus Connect (train radio)
- Open Data first made available ~2010, intended for “countdown clock”-type apps
- Multiple other developers have built live maps and other creative visualizations of the data



How does it work?

- An archiver polls the Trackernet API for each line every 30-45 seconds and stores the results in a ClickHouse database
- Materialized views in the database convert the API responses to useful arrivals data
- The frontend queries the database for arrivals and renders a plot with Vega-Lite

Challenges (1/2)

- Fundamentally, we are trying to form a train-wise view of the network from station-wise slices.
  - The line-level `PredictionSummary` API leaves out some of the fields in the station-level `PredictionDetailed` API, so we have to poll *both*, generating a lot of unnecessary load.
- Data quality is very good in CBTC territory, much less so in conventional territory, **especially** on the sub-surface railway.
- The Hammersmith & City line. 🤬
- Some trains cannot be uniquely identified (set number but no trip number). Also afflicts NR trips on the Bakerloo.
- The Trackernet API might be going away?

Challenges (2/2)

- Sometimes the Trackernet API just gives up and times out!
  - Particularly problematic for the `PredictionDetailed` endpoint with new line and station pairs.
  - `PredictionDetailed` also crashes sometimes when trains show up at stations where they do not belong.
- The `Location` field, which gets four whole slides of its own.
- Line-wise APIs return arrivals for other lines at the same station, leading to duplicates unless you filter them out.
  - Can’t rely on the LN field in the response; have to go by the set number and the set ranges published in the WTTs!
- Directions! (there is no trip-level direction field so we have to infer directionality based on which platforms a trip stops at…)

Things seen in the `Location` field

Location	Observations	Notes
`At Platform`	91,148,714	Nice and easy.
`At London Bridge Plaform 1`	1,374,804	“Plaform”?
`Left Angel, heading towards Kings Cross`	804,172	“heading towards”!? Why isn’t this just a “Between”?
`Between Notting Hill Gate and High Street Kensingt`	624,747	“Kensingt”? Did we hit a character limit somewhere?
`In between Wood Lane and Latimer Road`	331,768	“In between”?!
`Approachihng Belsize Park`	182,891	“Approachihng”?
`At Heathrow Terminal 1,2,3 Platform 1`	93,817	Terminals, plural.
`Upminster areao`	1,878	“areao”?
`Morden Platform 2`	162	At? Approaching?

But why does the `Location` field even matter?

- We want to be able to distinguish between *actual* observations and predicted arrivals, and the easiest way to do this is to parse the Location field.
- Can't we just look for "At Platform"?
  - No, sometimes we get "At" + station name or even "At" + station name + platform number.
- Therefore, a bunch of regular expressions…

xkcd.com/208

Several slightly gnarly regular expressions…

```
?P<location_type>Approachih?ng|Arriving At|At|Around|Departed|Departing|Leaving|Left|Near|
(?:North|South) of) (?P<location_name>(?:.(?!Depot))+?)(?:(?:,? (?:p|P))(?:lat?form)?(?: ?(?
P<platform_number>(?:[0-9 ]|(?:and))+))?)?$

^(?P<area_name>.+) [Aa]reao?(?: fast)?$

^(?:In b|B)etween (?P<from_station_name>.+) ((?<!Elephant )and) (?P<to_station_name>.+)$

^Left (?<from_station_name>.+)(?: Approaching |, heading to(?:wards)? )(?<to_station_name>.+)

^(?:(?:In|(?P<prefix>Siding at)) )?(?P<location_name>.+?)(?(<prefix>)| (?:S|s)idings?)(?: (?:Road |(?
P<lparen>\())(?P<road_number>.+?)(?(<lparen>)\)))?$

^(?:At )?(?P<depot_name>.+) Depot(?: entrance road)?$
```

And about 50 test cases…

```
#[rstest]
#[case("At Platform", AtPlatform)]
#[case("Approaching Mornington Crescent",
    Approaching{ location_name: "Mornington Crescent".to_string(), platform_number: None }
)]
#[case("Approachihng Belsize Park",
    Approaching{ location_name: "Belsize Park".to_string(), platform_number: None }
)]
#[case("Approaching  Burnt Oak",
    Approaching{ location_name: "Burnt Oak".to_string(), platform_number: None }
)]
#[case("Approaching Battersea Power Station Platform 1",
    Approaching{ location_name: "Battersea Power Station".to_string(), platform_number:
Some("1".to_string()) }
)]
#[case("Approaching Edgware Road Platform 1 and 2",
    Approaching{ location_name: "Edgware Road".to_string(), platform_number: Some("1 and
2".to_string()) }
)]
#[case("Arriving At Chorleywood",
    Approaching{ location_name: "Chorleywood".to_string(), platform_number: None }
)]
#[case("Departed Euston",
    Departed{ location_name: "Euston".to_string(), platform_number: None }
)]
#[case("Departing Pimlico",
    Departing{ location_name: "Pimlico".to_string(), platform_number: None }
)]
#[case("Departing East Finchley, platform 1",
    Departing{ location_name: "East Finchley".to_string(), platform_number: Some("1".to_string()) }
)]
#[case("Leaving Green Park",
    Leaving{ location_name: "Green Park".to_string(), platform_number: None }
)]
#[case("Left Angel Platform 1",
    Left{ location_name: "Angel".to_string(), platform_number: Some("1".to_string()) }
)]
#[case("Left Amersham", Left{ location_name: "Amersham".to_string(), platform_number: None })]
#[case("Near Chesham", Near{ location_name: "Chesham".to_string(), platform_number: None })]
#[case("North of Piccadilly Circus",
    NorthOf{ location_name: "Piccadilly Circus".to_string(), platform_number: None }
)]
#[case("South of Oxford Circus",
    SouthOf{ location_name: "Oxford Circus".to_string(), platform_number: None }
)]
#[case("At Shepherds Bush Market Platform 1",
    At{ location_name: "Shepherds Bush Market".to_string(), platform_number: Some("1".to_string()) }
)]
#[case("At Snaresbrook", At{ location_name: "Snaresbrook".to_string(), platform_number: None })]
#[case("Around Mill Hill East",
    Around{ location_name: "Mill Hill East".to_string(), platform_number: None }
)]
#[case("Left Angel, heading towards Kings Cross",
    Between{ from_station_name: "Angel".to_string(), to_station_name: "Kings Cross".to_string() }
)]
#[case("Left West Finchley, heading to Woodside Park",
    Between{ from_station_name: "West Finchley".to_string(), to_station_name: "Woodside
Park".to_string() }
)]
```

```
#[case("Ealing Broadway area", Area{ area_name: "Ealing Broadway".to_string() })]
#[case("Brixton Area", Area{ area_name: "Brixton".to_string() })]
#[case("Neasden Area fast", Area{ area_name: "Neasden".to_string() })]
#[case("Upminster areao", Area{ area_name: "Upminster".to_string() })]
#[case("Between Heathrow Terminal 5 and Heathrow Terminal 1,2,3",
    Between{ from_station_name: "Heathrow Terminal 5".to_string(), to_station_name: "Heathrow Terminal
1,2,3".to_string() }
)]
#[case("Between Elephant and Castle and Kennington",
    Between{ from_station_name: "Elephant and Castle".to_string(), to_station_name:
"Kennington".to_string() }
)]
#[case("Between Lambeth and Elephant and Castle",
    Between{ from_station_name: "Lambeth".to_string(), to_station_name: "Elephant and
Castle".to_string() }
)]
#[case("WERD", Other)]
#[case("At Angel platform", At{ location_name: "Angel".to_string(), platform_number: None })]
#[case("In between Goldhawk Road and Hammersmith",
    Between{ from_station_name: "Goldhawk Road".to_string(), to_station_name:
"Hammersmith".to_string() }
)]
#[case("Left Queen's Park Approaching Network Rail Track",
    Between{ from_station_name: "Queen's Park".to_string(), to_station_name: "Network Rail
Track".to_string() }
)]
#[case("Departing East Finchley, platform 1",
    Departing{ location_name: "East Finchley".to_string(), platform_number: Some("1".to_string()) }
)]
#[case("At East Finchley, Platform 1",
    At{ location_name: "East Finchley".to_string(), platform_number: Some("1".to_string()) }
)]
#[case("Lillie Bridge Depot", Depot{ depot_name: "Lillie Bridge".to_string() })]
#[case("London Road Depot entrance road", Depot{ depot_name: "London Road".to_string() })]
#[case("Acton Sidings", Sidings{ location_name: "Acton".to_string(), road_number: None })]
#[case("Edgware Road Siding Road 26",
    Sidings{ location_name: "Edgware Road".to_string(), road_number: Some("26".to_string()) }
)]
#[case("In Hammersmith Sidings",
    Sidings{ location_name: "Hammersmith".to_string(), road_number: None }
)]
#[case("In Hammersmith sidings (P24)",
    Sidings{ location_name: "Hammersmith".to_string() , road_number: Some("P24".to_string()) }
)]
#[case("Siding at Finchley Central",
    Sidings{ location_name: "Finchley Central".to_string(), road_number: None }
)]
#[case("At Hainault Depot", Depot{ depot_name: "Hainault".to_string() })]
#[case("At Kings Cross P7",
    At{ location_name: "Kings Cross".to_string(), platform_number: Some("7".to_string()) }
)]
#[case("At London Bridge Plaform 1",
    At{ location_name: "London Bridge".to_string(), platform_number: Some("1".to_string()) }
)]
fn parser_test(#[case] input: &str, #[case] expected: LocationType) {
    assert_eq!(expected, parse(input))
}
```

Obligatory disclaimer: these challenges are discussed in good faith. We're not here just to beat up on agencies who are often trying their hardest in challenging situations.

Demo!

trackernet-stringlines.choochoo.systems

kurtraschke.com/2025/06/trackernet-stringlines