Population Based Routing in LEO Satellite Networks

dr. Anders Fongen, June 2022
Norwegian Defence University College, Cyber Defence Academy, Lillehammer
email: anders@fongen.no

MOBILITY2022, Porto, Portugal

Presenter's bio

Anders Fongen

- Associate Professor, Norwegian Defence University College
- Field of research: Distributed Systems, Networking security
- PhD in Distributed Systems, Univ. of Sunderland, UK, 2004
- Career history
  - 5 years in military engineering education
  - 10 years research in defence research (Chief Scientist)
  - 8 years in civilian college (Associate professor)
  - 11 years in oil industry
  - 6 years in electronics industry

Introduction

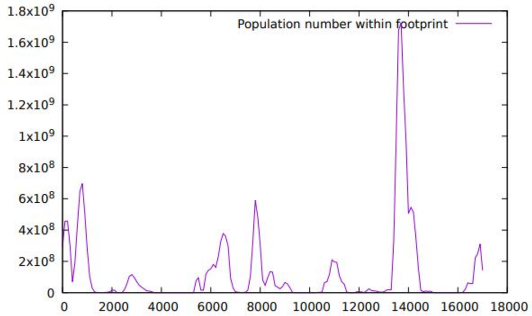
- The evolution of satellite communication?
  - Application Services (“Cloud Computing in Space”)
  - Higher System Complexity (larger state space)
- What are the advantages?
  - Very Low Latency (as low as 3 ms)
  - Global coverage
- Interesting property of a Low Earth Orbit (LEO) system
  - Long idle periods (due to inhabited surface) mixed with traffic peaks
- Viewed as a problem of *Distributed Computing*
  - *having a set of distinct properties*

What is a SIN (Space Information Network)?

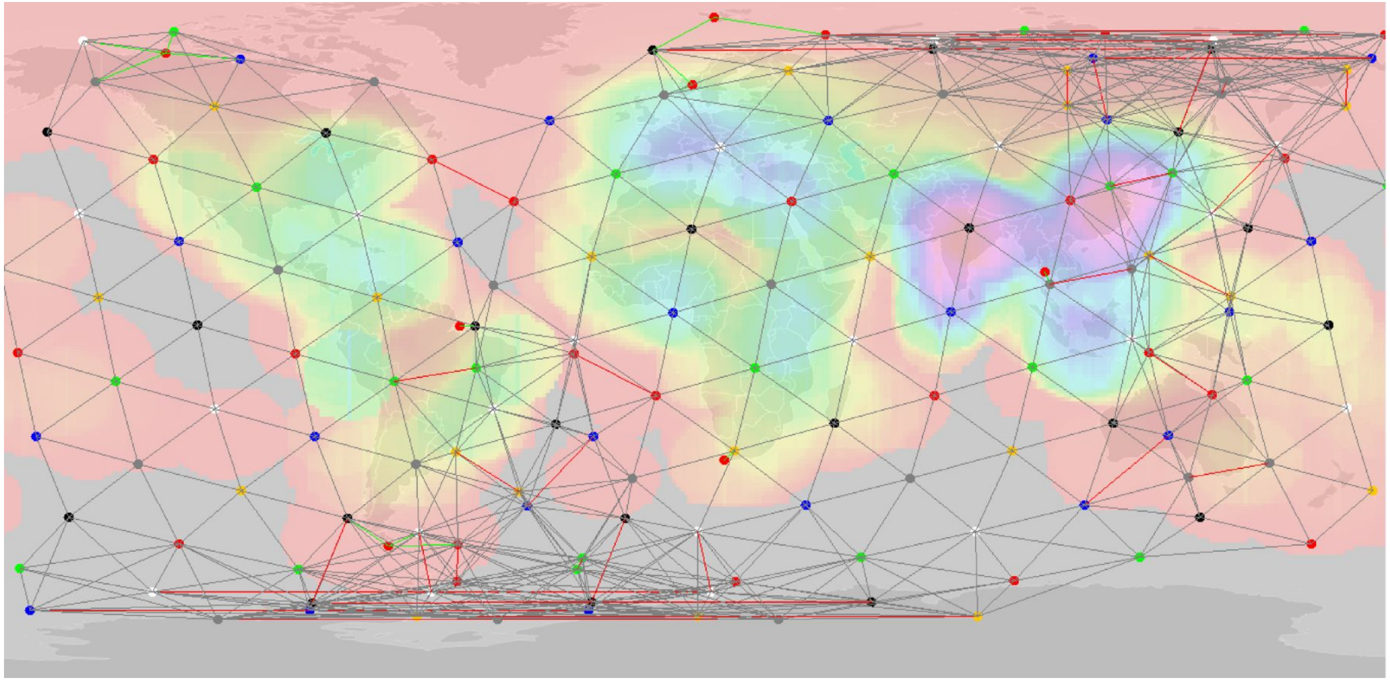
- A collection of communicating LEO satellites
- Able to serve terrestrial/airborne client
  - Communication services (e.g., IP transport, VoIP, Publish-Subscribe comm.)
  - Discovery Services (DNS, Service Brokering...)
  - Storage Services (Content Distribution Network, caching, session states)
  - Application Services (Collaborating editing, Situational awareness ...)
- Resource constrained / disadvantaged
- Predictable workload and link availability
- “Mobile” system: Stationary clients, mobile infrastructure
- Rapid hand-over of client connection and *client state*

Why using demographic data in routing?

- A global satellite system will seldom be scaled to a global population
  - Due to high population density in small areas
- During orbit, a satellite is idle most of the time
  - During which it can offload the busy satellites, like ***routing***
- We choose to calculate route cost based on receiver ***footprint population***
  - Can be calculated, based on time, date and orbital elements
- Experimental results will follow....



Population “heat map” from satellite footprint



Four different routing methods compared

- Baseline
  - Hot Potato - “throw in best direction”
  - Unweighted Dijkstra - disregard population density

- Experimental
  - Population based Dijkstra - link cost derived from population density
  - Delayed forwarding - wait for a better route to appear

Hot potato routing

“Throw in best direction”:

- Each forwarding node calculate the bearing to the final destination
- Picks the link with the direction closest to the bearing
- Result: Effective when there is a path
  - otherwise cycling and packet loss
  - TTL was needed

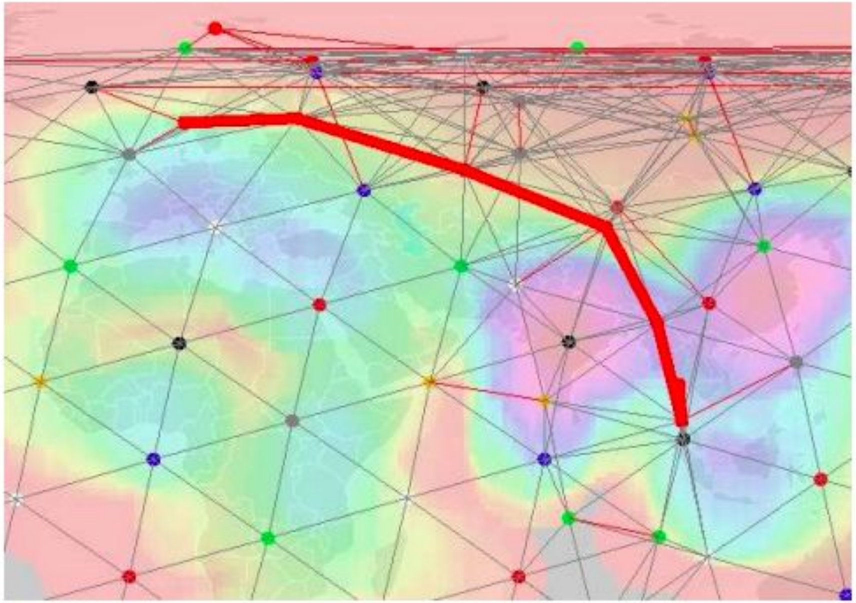


Figure 3. Example result from the hot potato routing algorithm.

Unweighted Dijkstra

All links having equal weight, minimizes the number of hops

- Source routing was chosen, so no packet is sent unless there exists a path
  - TTL not needed
- Chooses routes overlooked by the hot potato routing, e.g., over polar regions

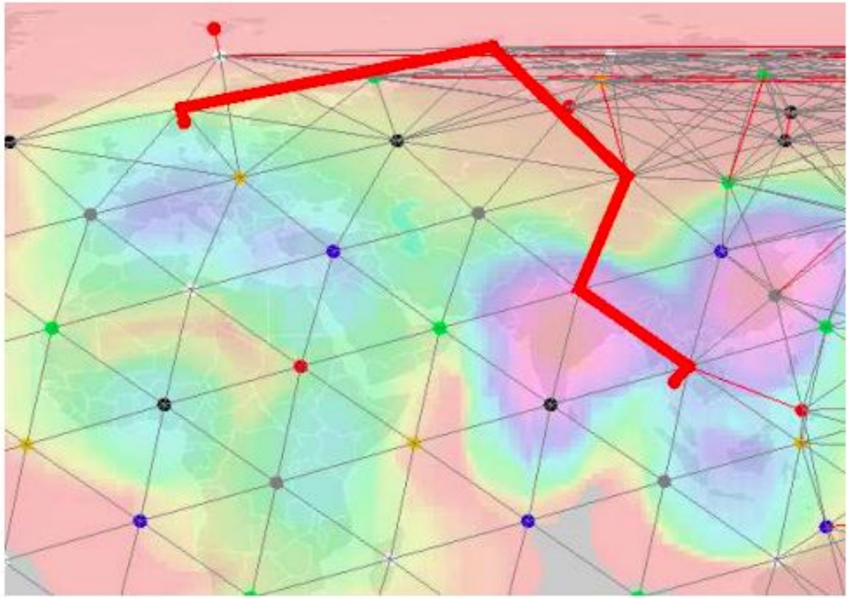


Figure 4. Example result from the unweighted Dijkstra's routing algorithm.

Population based Dijkstra

Link weight derived from receiver's *footprint population*

- Minimizes the population number along the path,
  - avoiding the busiest satellites
  - creates longer paths

- Employs otherwise unused transmission resources (hopefully)

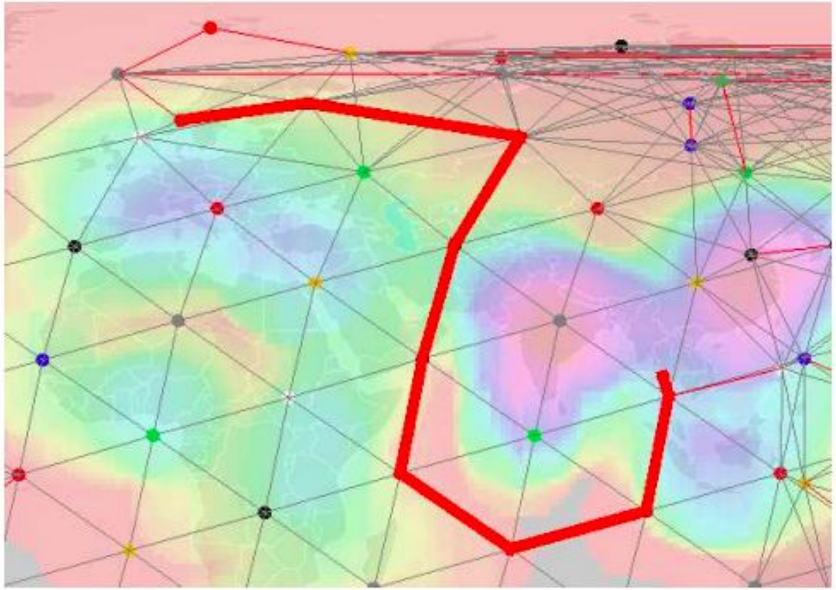
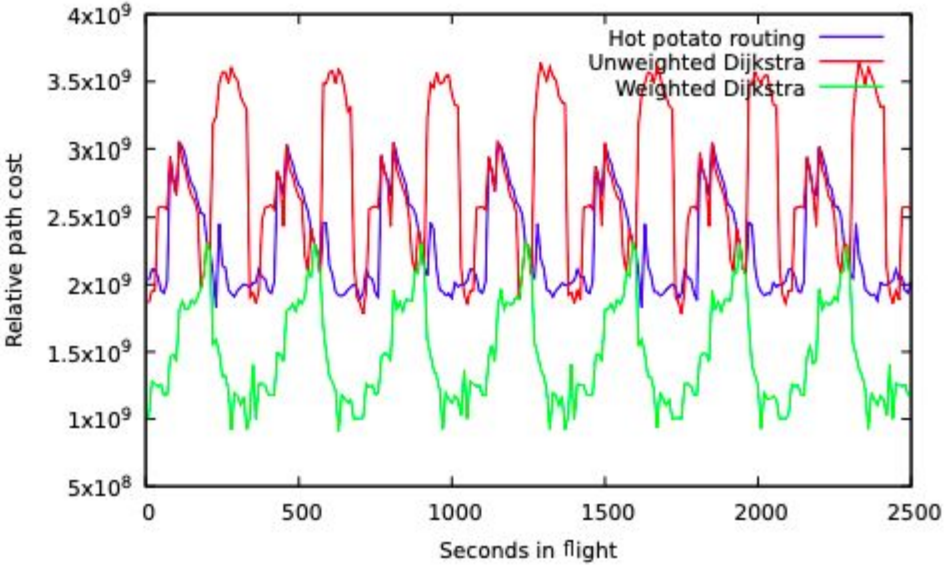


Figure 5. Example result from the weighted Dijkstra's routing algorithm.

Routing methods compared

Comparing the 3 methods with regard to ***population along path:***

- "Unweighted Dijkstra" has high maximums
- "Population based" has nice minimums
- "Hot potato" is something in between the other two



Observe the strong cyclic properties!

Proposing a “delay tolerant” routing

With regard to the cyclic properties of the path cost: Wait for a “cheap” path to appear

- Calculate cost distribution over time
- When sending a packet, wait until path cost is lower than a given percentile
- Observe the path cost vs queueing time
- Conclusion: A reduction is observed, **but not worth the delay**

TABLE I
RESULTING QUEUING DELAY AND ROUTING COST WHEN DIFFERENT PERCENTILES ARE USED AS SENDING THRESHOLDS

Percentile	Queuing delay (secs)	Routing cost ($\cdot 10^7$)
10	855.92	99.35
20	544.46	105.08
30	91.38	111.11
40	53.79	116.63
50	29.14	119.41
60	12.98	123.83
70	5.78	126.79
80	2.83	127.98
90	0.65	130.05

Conclusion

- Population based routing shows a reduction in average path cost on approx. 40-50 % over unweighted Dijkstra.
    - *This is the result of our hypothesis*
- Both methods shows strong cyclic properties, and path cost is highly dependent on the momentary satellite positions.
- Path cost variations can be predicted and the best moment for transmission can be calculated. This is probably useful and wirth the extra delay

- Internet penetration, time of day, day of week etc. can be included in the link cost calculations