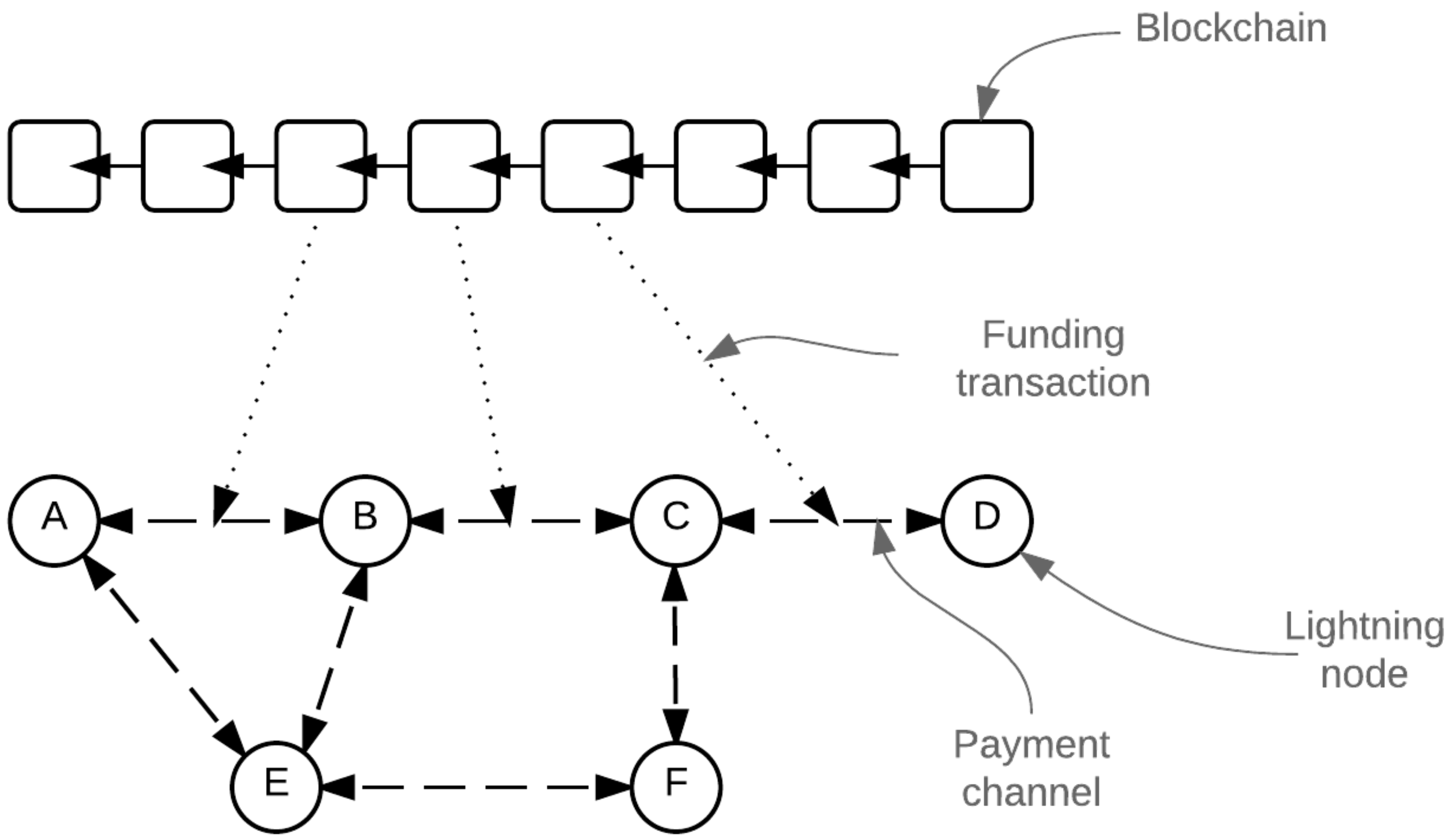
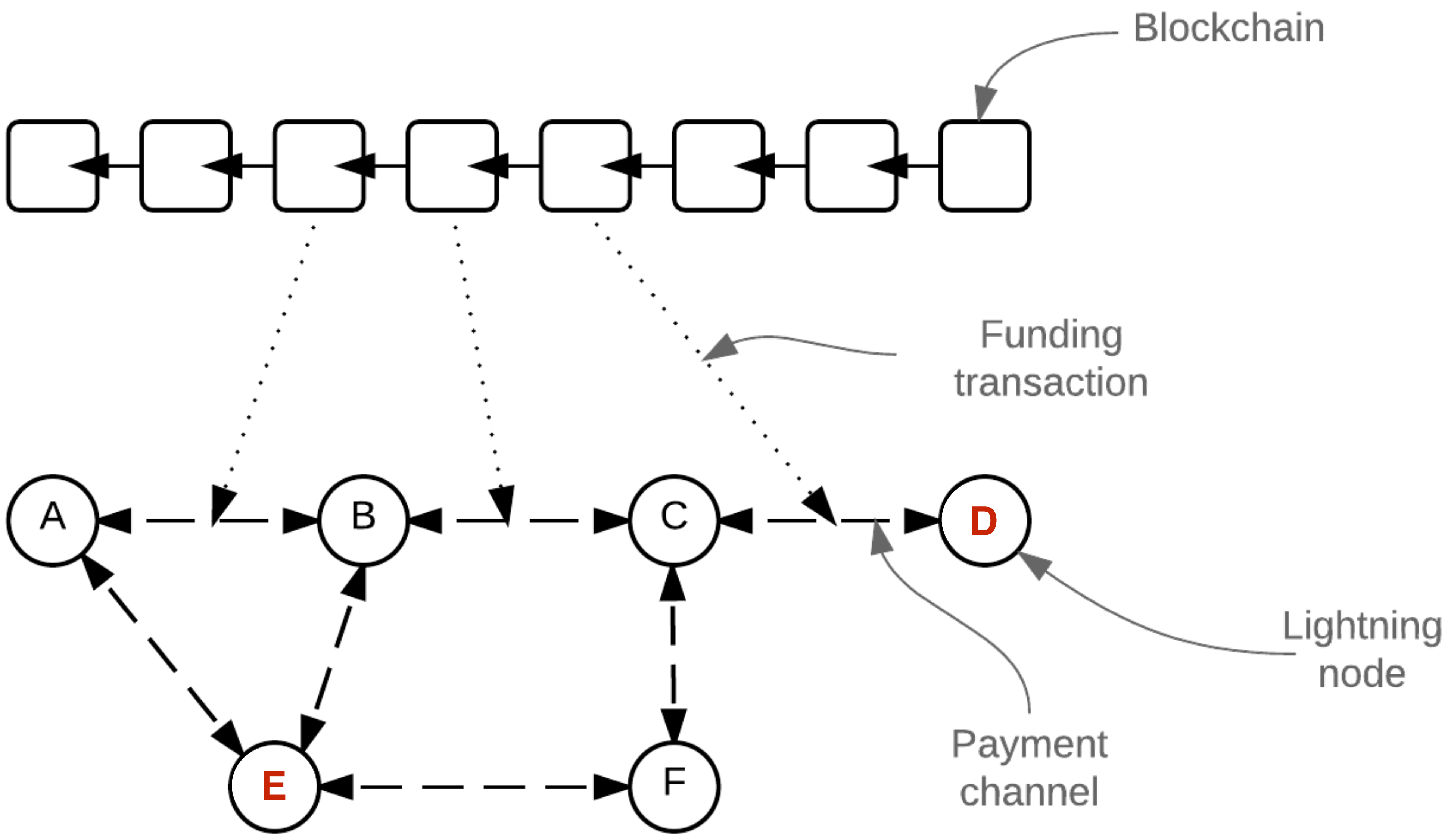
Flare: An approach to routing in Lightning Network

Pavel Prihodko
Kolya Sakhno
Alexei Ostrovskiy
Slava Zhigulin
Olaoluwa Osuntokun

Lightning network



Routing



How E finds path to D ?

Routing requirements

- Peer-to-peer network
- Source routing
- Trustlessness
- Anonymity
- Fast payment processing

Routing

As a solution we came up with algorithm **Flare**:

http://bitfury.com/content/5-white-papers-research/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf

Core idea

State of LN can be split in two distinct components:

- Payment channels,
- Total capacity,

← *Slowly changing, static information*

- Status of nodes,
- Distribution of funds,
- Fees for using a channel.

← *Quickly changing, dynamic information*

Flare design

Proactive part *(on schedule):*

- Gather static information - *store open channels*

Reactive part *(on payment request):*

- Gather dynamic information - *ask funds, fees, status*
- Find path based on both

Proactive

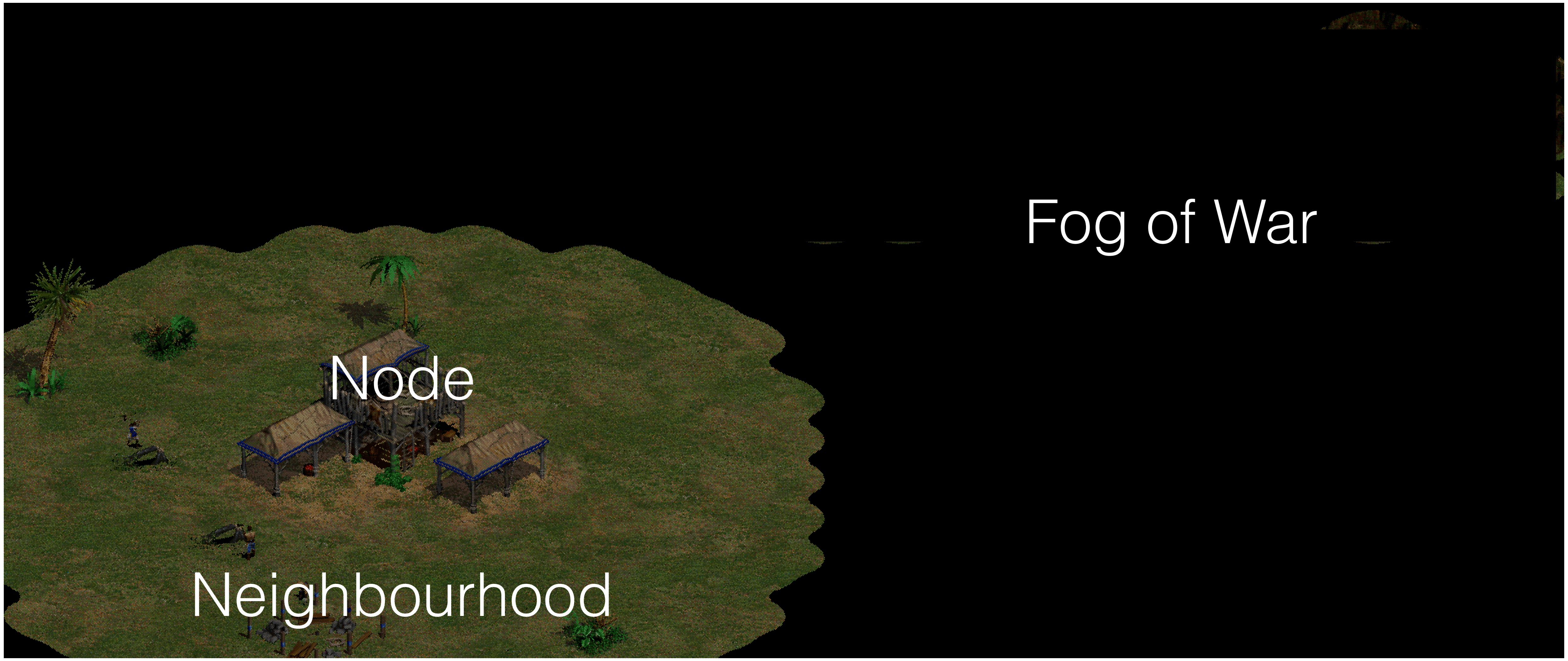
Proactive: Neighbourhood

In certain radius node can very quickly gather information on channels/opening closing, thus having up to date picture, but it is not scalable to have radius too big

Neighbours

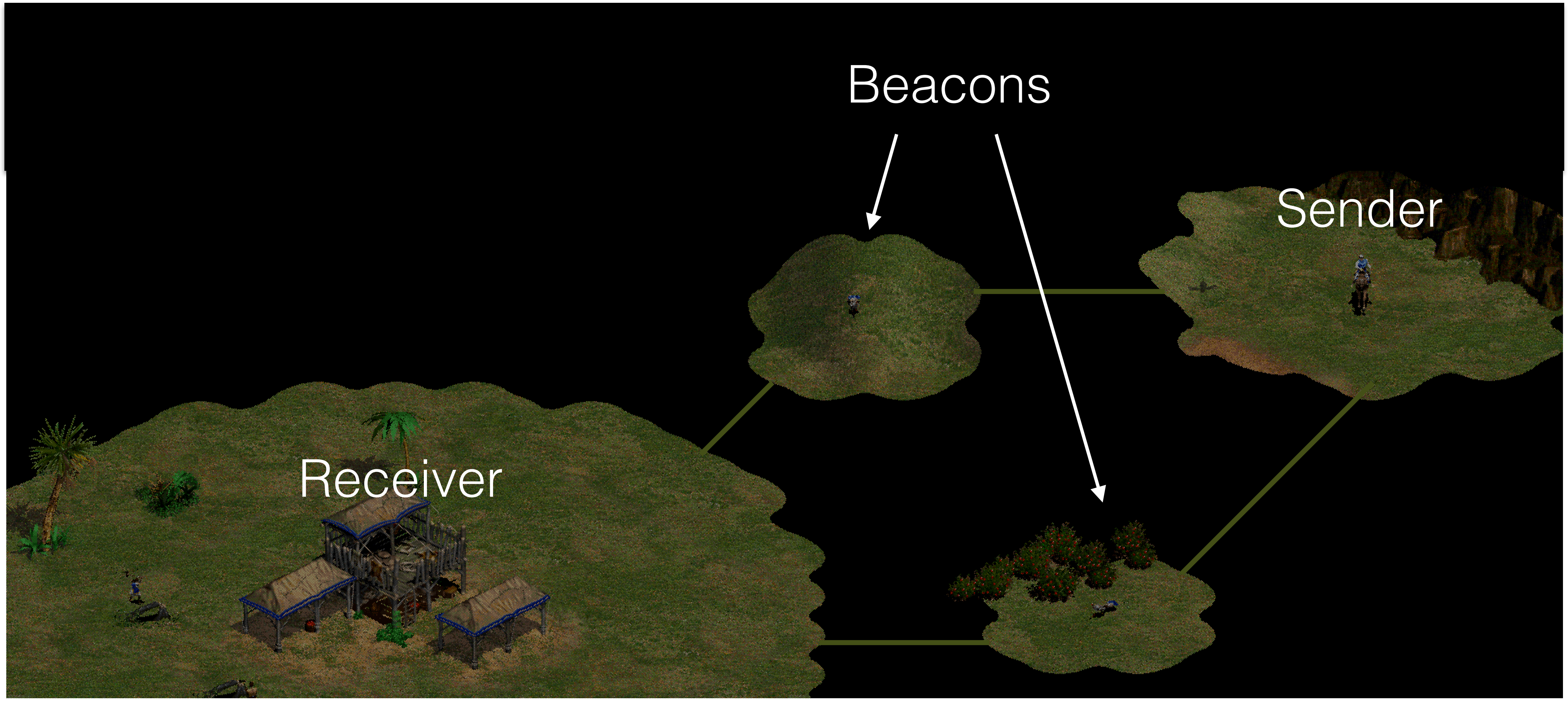
- Each node propagates information on it’s channels closing and opening in certain radius
- This allows each node to have up to date picture of all open channels (with their total capacity) in certain radius.

Proactive



What to do if network becomes huge?

Proactive: Beacons

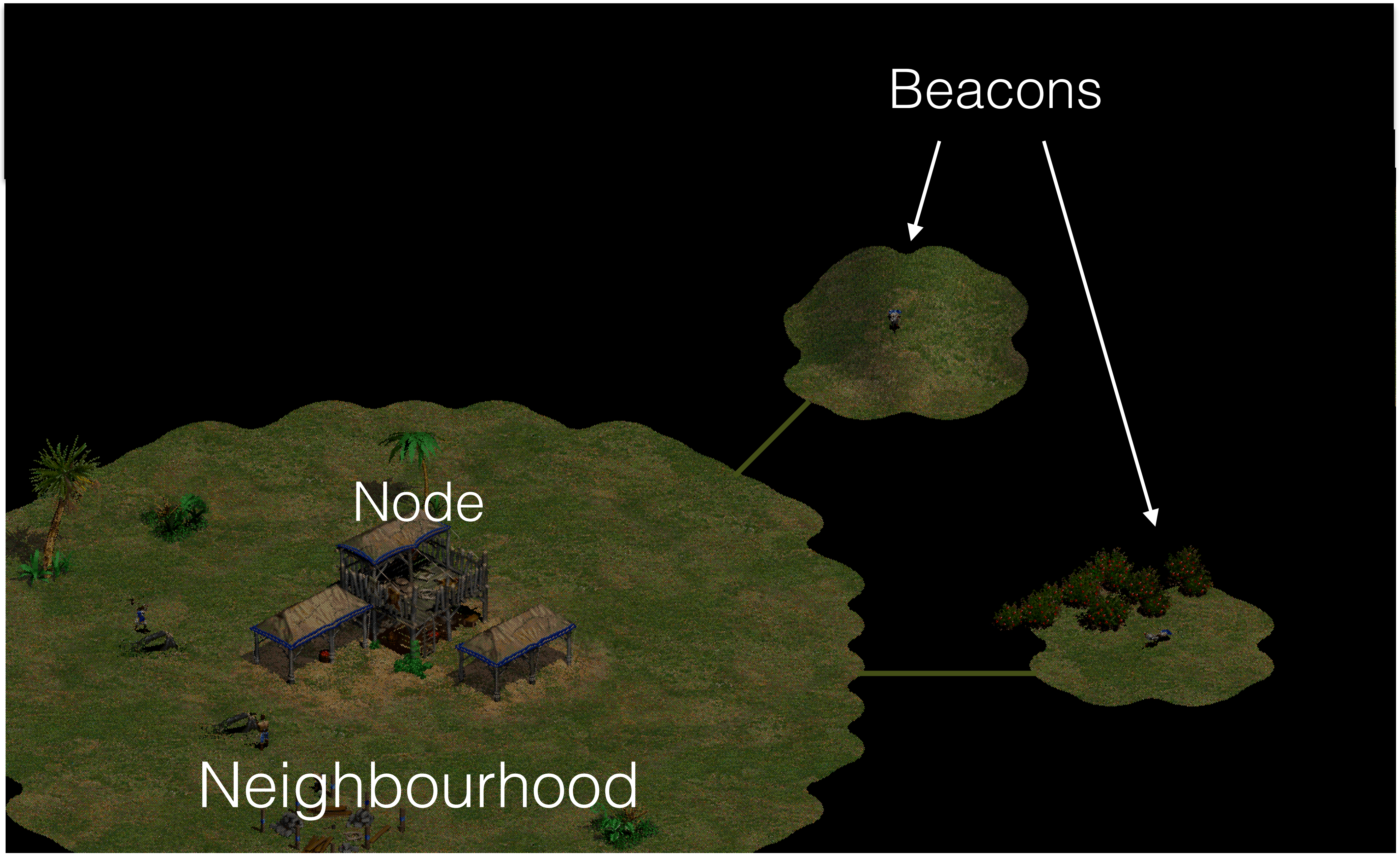


To enhance long range visibility node finds paths to distant nodes (beacons) which can help to find route to receiver if he is not in the neighbourhood

Beacons

- Each node finds paths to nodes whose addresses are closest to the one’s (claiming them beacons)
- On reactive stage this allows to search for longer paths iterating over known nodes in DHT like manner

Routing Table



Reactive

Reactive

When node E wants to send money to D:

1. E and D find path candidates on the graph of their routing tables

2. If no candidates are found E requests tables from nodes whose addresses are closest to D and so on…

3. When several candidates are found E collects dynamic information on them

4. If the one is found E creates HTLC and sends money to D

Reactive

When node E wants to send money to D:

1. **E and D find path candidates on the graph of their routing tables**

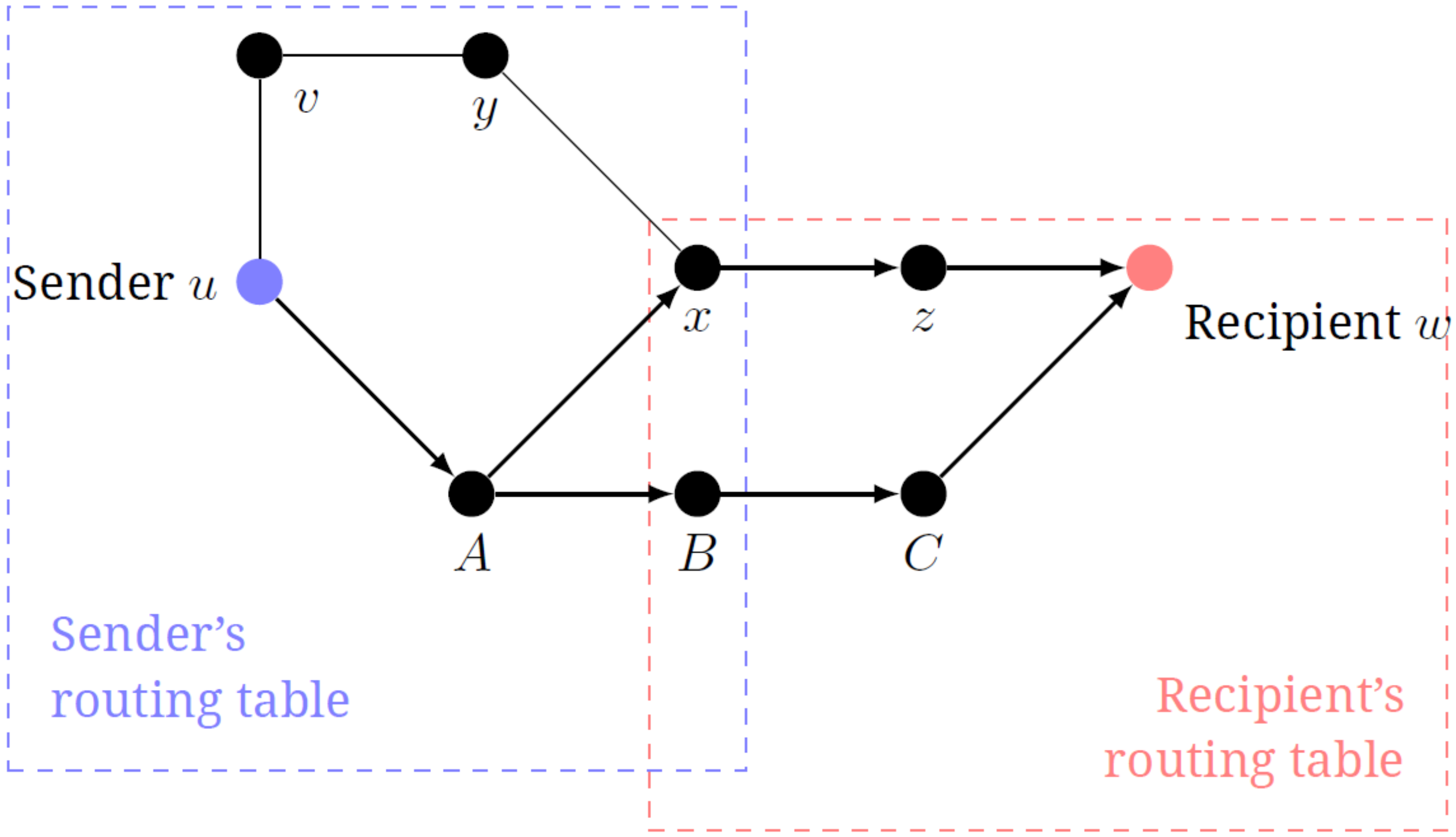
2. If no candidates are found E requests tables from nodes whose addresses are closest to D and so on…

3. When several candidates are found E collects dynamic information on them

4. If the one is found E creates HTLC and sends money to D

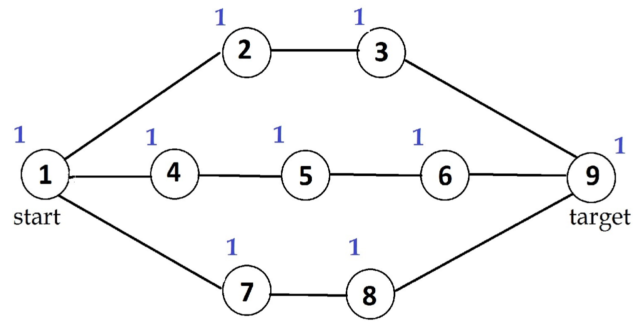
Finiding candidates

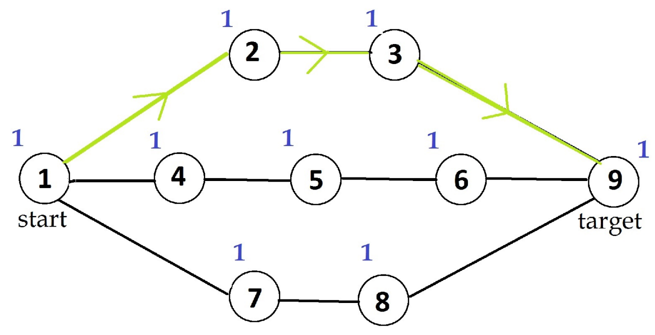
Once joint routing table is created one may find k paths using approaches like breadth-first search

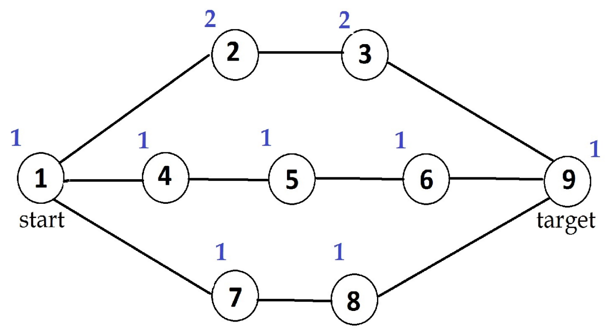


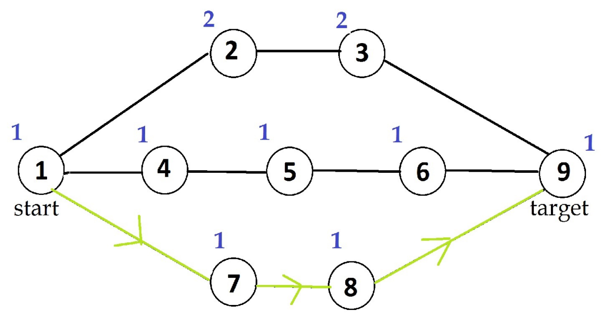
Disjoint paths

By adding vertex weights to network graph one may find shortest paths that are most different from previous found

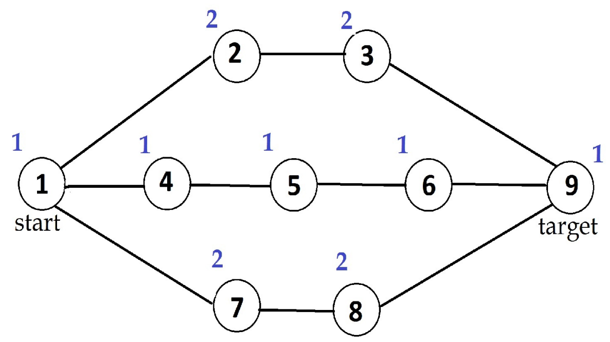


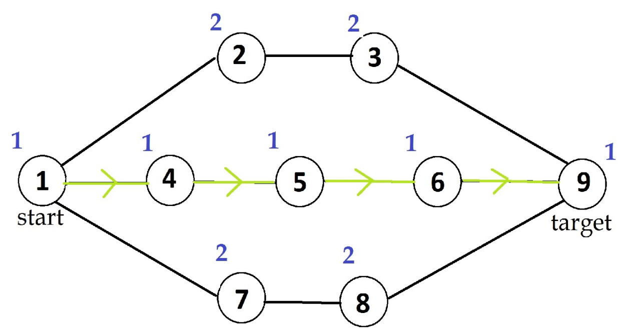


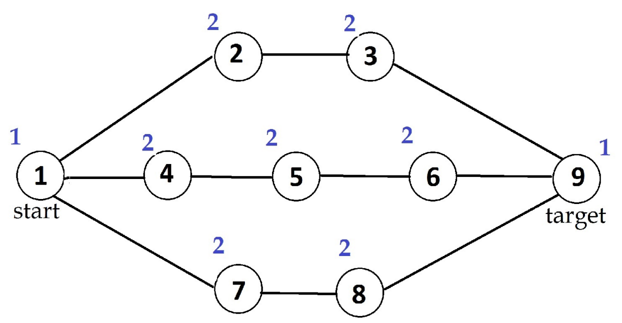


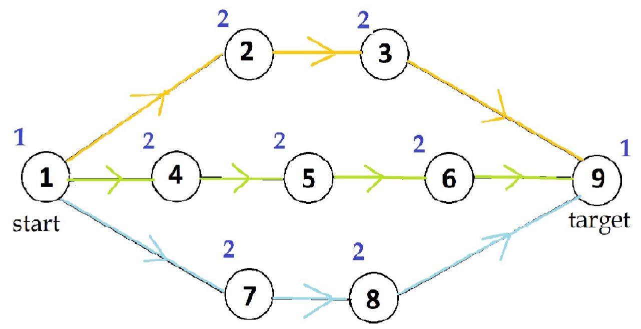


Initial graph $\longrightarrow$ Shortest path 1 $\longrightarrow$ Weights update $\longrightarrow$ Shortest path 2 . . .









. . . Weights update $\longrightarrow$ Shortest path 3 $\longrightarrow$ Weights update $\longrightarrow$ Found paths

Reactive

When node E wants to send money to D:

1. E and D find path candidates on the graph of their routing tables

2. **If no candidates are found E requests tables from nodes whose addresses are closest to D and so on…**

3. When several candidates are found E collects dynamic information on them

4. If the one is found E creates HTLC and sends money to D

Reactive

When node E wants to send money to D:

1. E and D find path candidates on the graph of their routing tables

2. If no candidates are found E requests tables from nodes whose addresses are closest to D and so on…

3. **When several candidates are found E collects dynamic information on them**

4. If the one is found E creates HTLC and sends money to D

Dynamic data

- Found candidates are paths that **potentially** can route the payment
- To tell if there is the path that we can use to route the payment we need to gather dynamic data for candidates *(funds, fees)*
- The simple solution - **probing onion messages** that traverse through candidate paths and quickly collect dynamic information

Dynamic data

But how do we know which candidates to check first? Need ranking.

- Distribution of funds in the channel - **uniform** *if know nothing*

- Probability that channel with capacity C would be able to route the payment x *is equal to* $max\left(0, 1 - \frac{x}{C}\right)$

- Probability payment x would make it through is

$$P\left(x \mid path\right) = \prod_{i \in path} max\left(0, 1 - \frac{x}{C_i}\right)$$

- After we get the probabilities we can start sending probes through the candidates with highest chance of success

Reactive

When node E wants to send money to D:

1. E and D find path candidates on the graph of their routing tables

2. If no candidates are found E requests tables from nodes whose addresses are closest to D and so on…

3. When several candidates are found E collects dynamic information on them

4. **If the one is found E creates HTLC and sends money to D**

Implementations

Two implementations of LN with Flare *(work in progress):*

- https://github.com/LightningNetwork/lnd
- https://github.com/ACINQ/eclair

What the real topology of LN would be ???

There is still no sensible topology and behavioural model of network and we need it for better experiments and fine tuning.

Thank you

Contact mail
lightning@bitfury.com