Advancing Schedule-Aware Bundle Routing for Space Delay-Tolerant Networks

NTN Days 2025

Olivier De Jonckère (LIRMM) Supervisor
Juan A. Fraire (INRIA) Supervisor

Longrui Ma (INRIA)

October 2025

Table of Contents

01 Delay-Tolerant Networking

02 SABR/CGR and challenges

03 A-SABR Library

04 Evaluation

05 Conclusion and Future works

1st Part

Delay-Tolerant Networking

Delay-Tolerant Networks and Bundle Protocol

Intermittent connectivity

- Planetary or spacecraft occlusion

End-to-end delays

- Large interplanetary distances
- Disruptions

Asymmetric bandwidths
High failure rates

- Power constraints
- Antenna pointing issues

To support large data units:

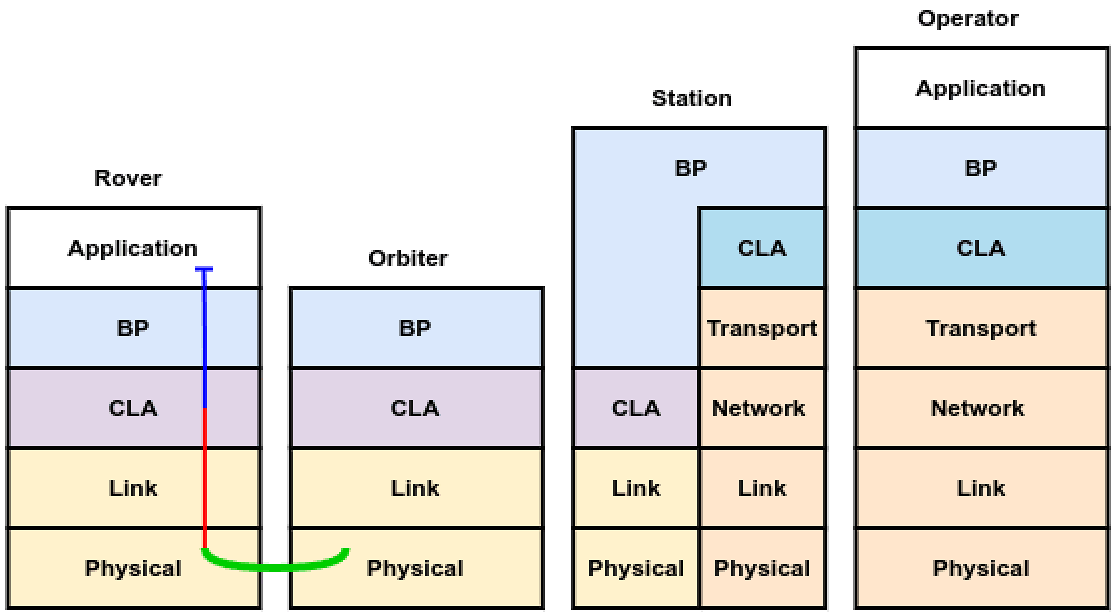
Delay-Tolerant Networking Architecture [1]

- store-and-forward

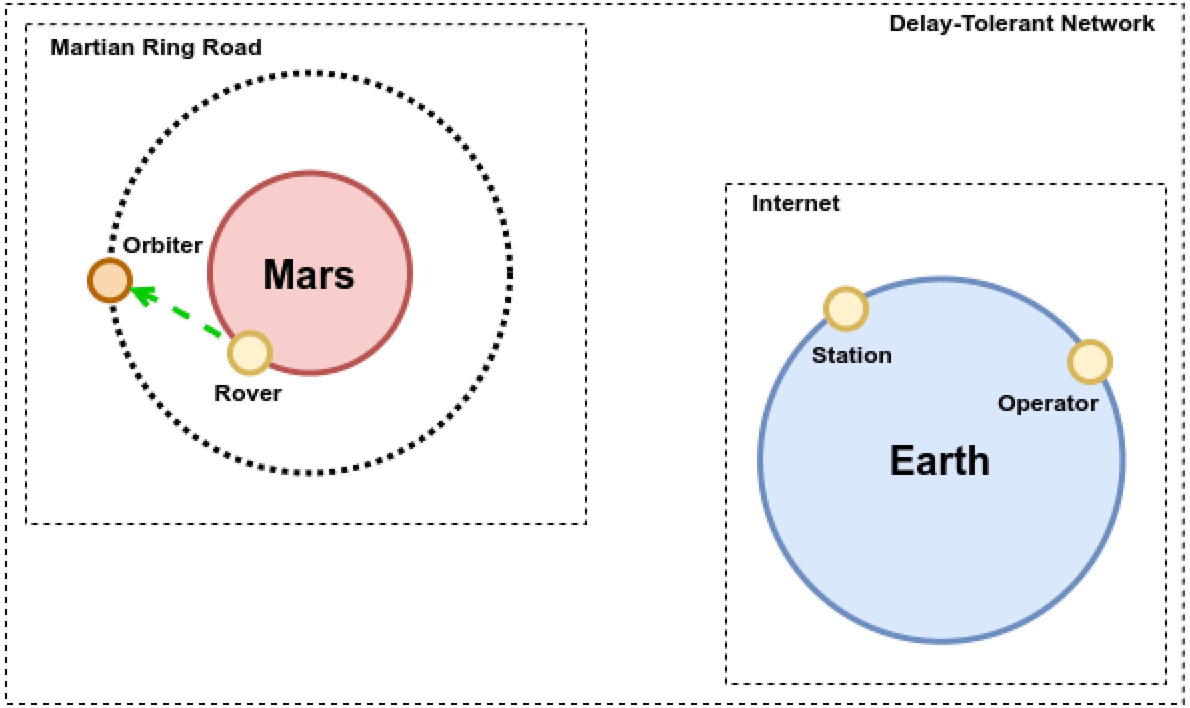
Bundle Protocol [2]

- end-to-end addressing and message encapsulation

Delay-Tolerant Networks and Bundle Protocol



CLA Convergence Layer Adapter BP Bundle Protocol

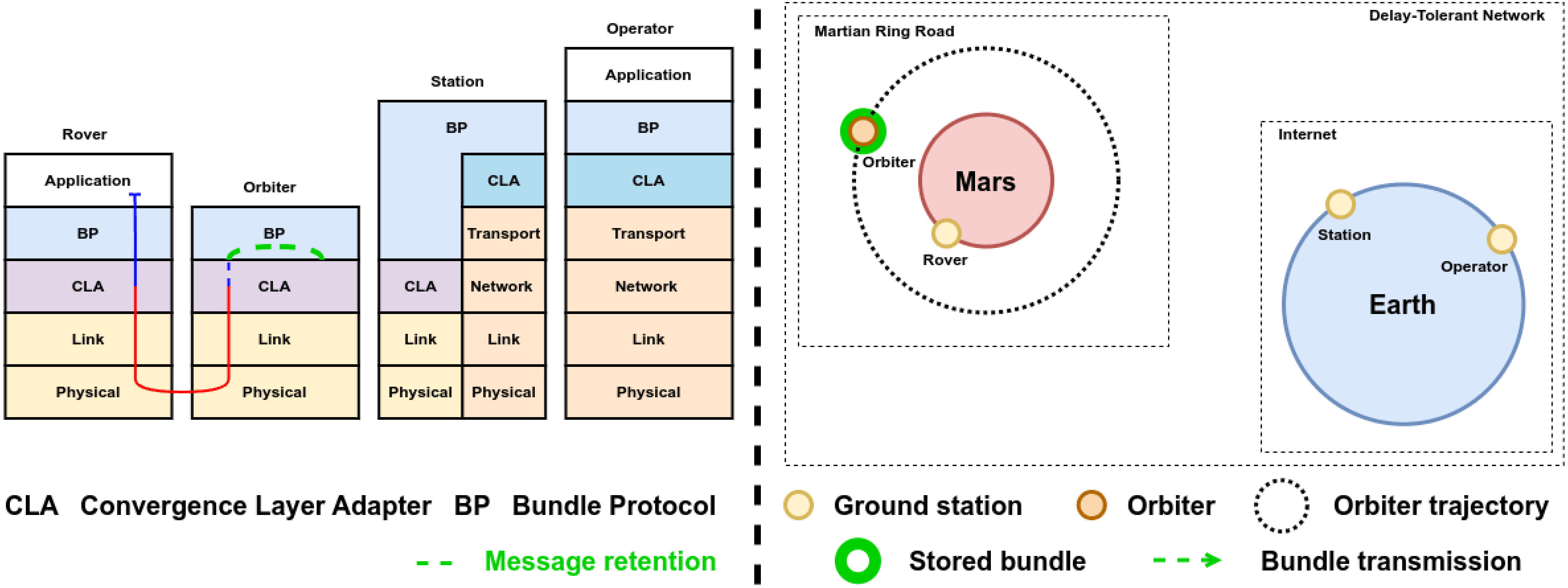


○ Ground station ● Orbiter ◌ Orbiter trajectory

⇢ Bundle transmission

Bundle sent from Rover to Obiter

Delay-Tolerant Networks and Bundle Protocol



CLA Convergence Layer Adapter BP Bundle Protocol

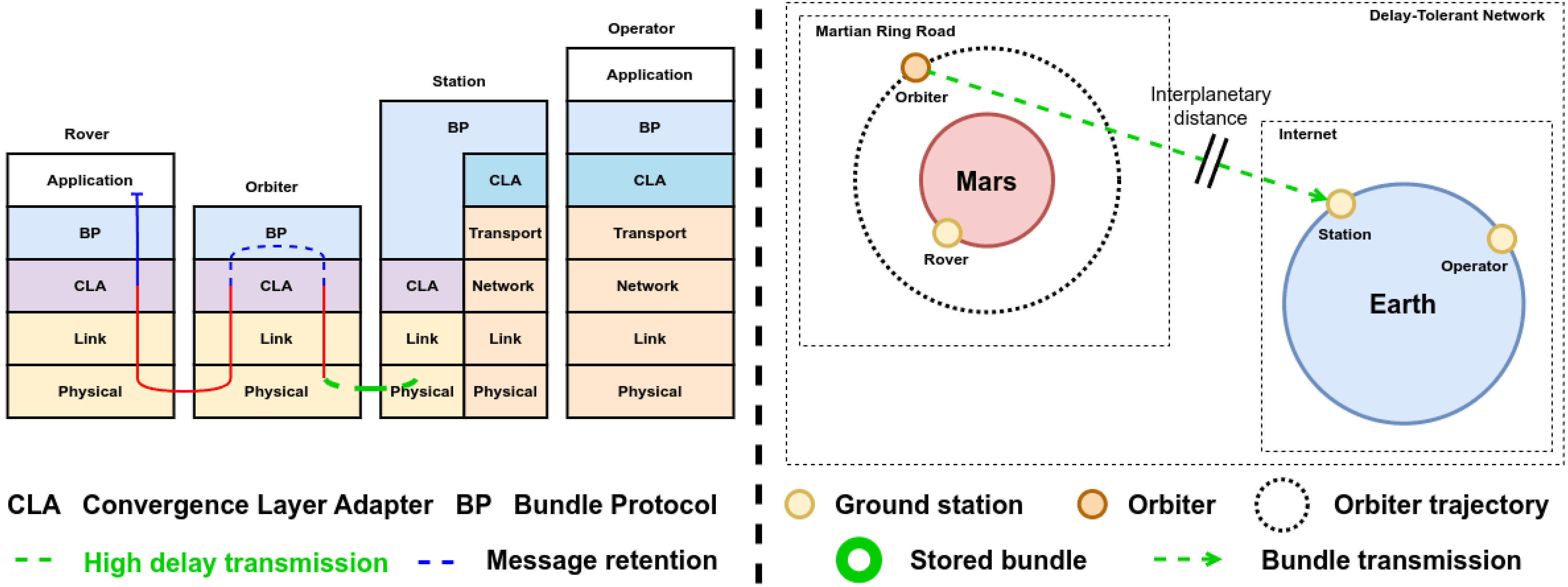
- - Message retention

○ Ground station ● Orbiter ◌ Orbiter trajectory

◎ Stored bundle - - ➔ Bundle transmission

Bundle stored in Obiter

Delay-Tolerant Networks and Bundle Protocol



CLA Convergence Layer Adapter BP Bundle Protocol

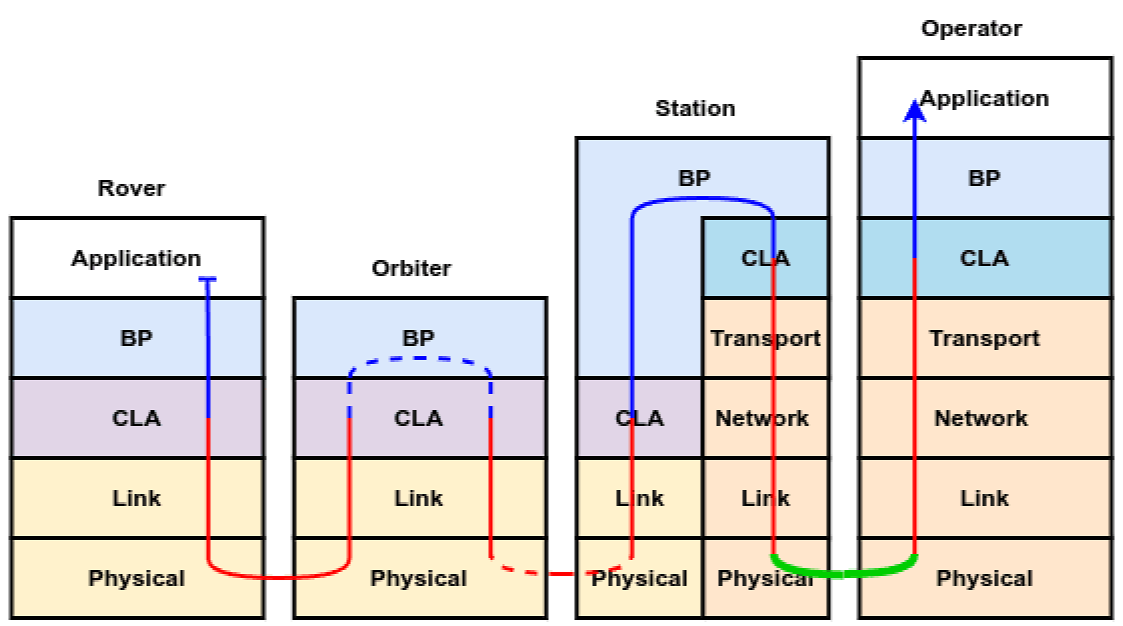
- - High delay transmission - - Message retention

○ Ground station ● Orbiter ◌ Orbiter trajectory

◎ Stored bundle - - → Bundle transmission

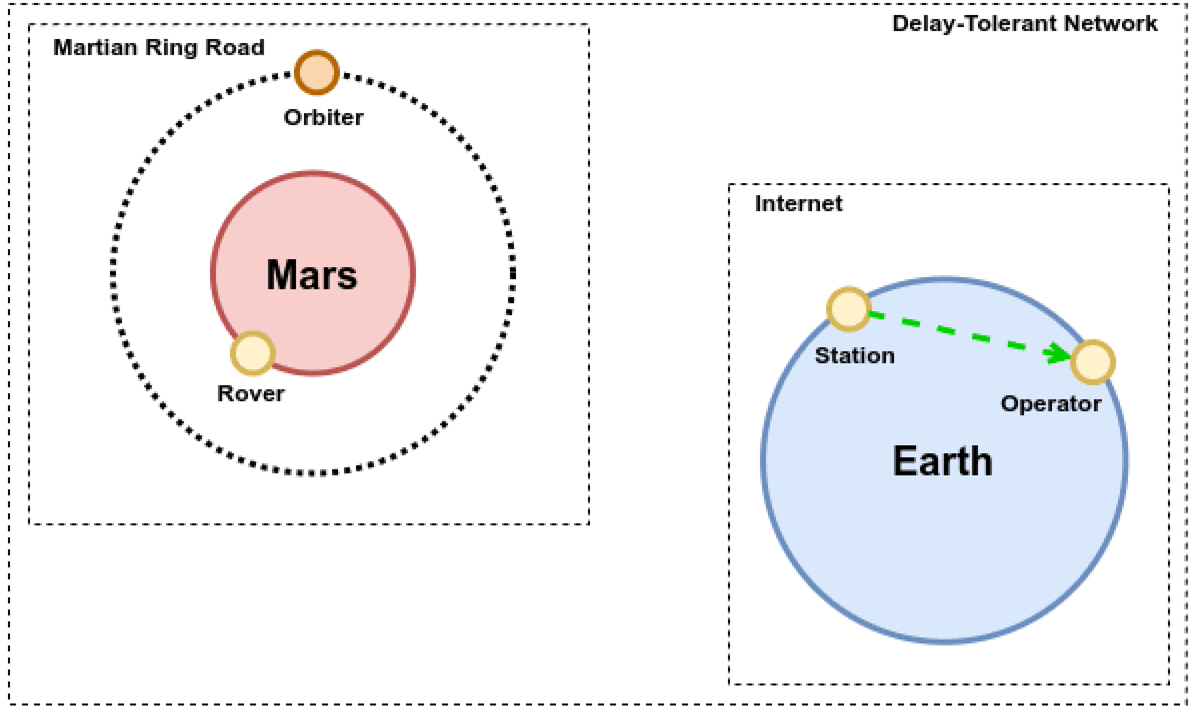
Bundle forwarded from Obiter to Station on Earth

Delay-Tolerant Networks and Bundle Protocol



CLA Convergence Layer Adapter BP Bundle Protocol

🟥 High delay transmission 🟦 Message retention

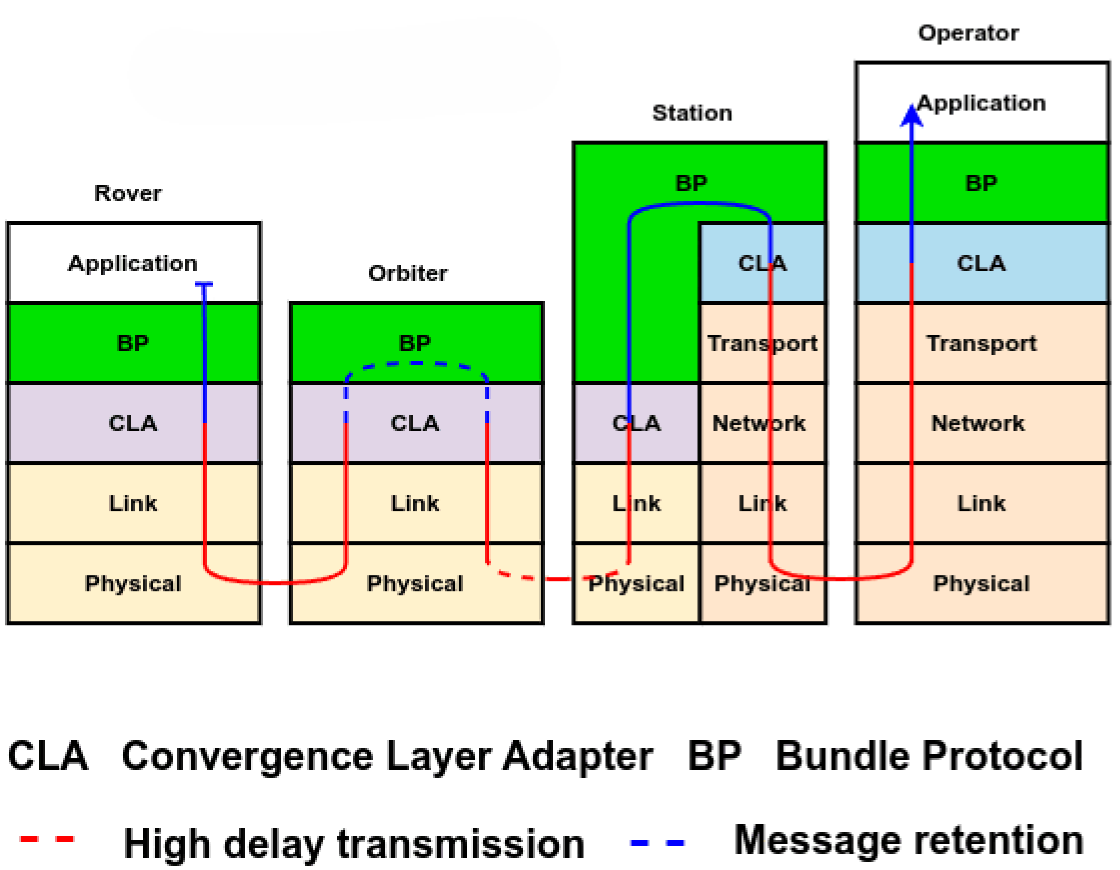


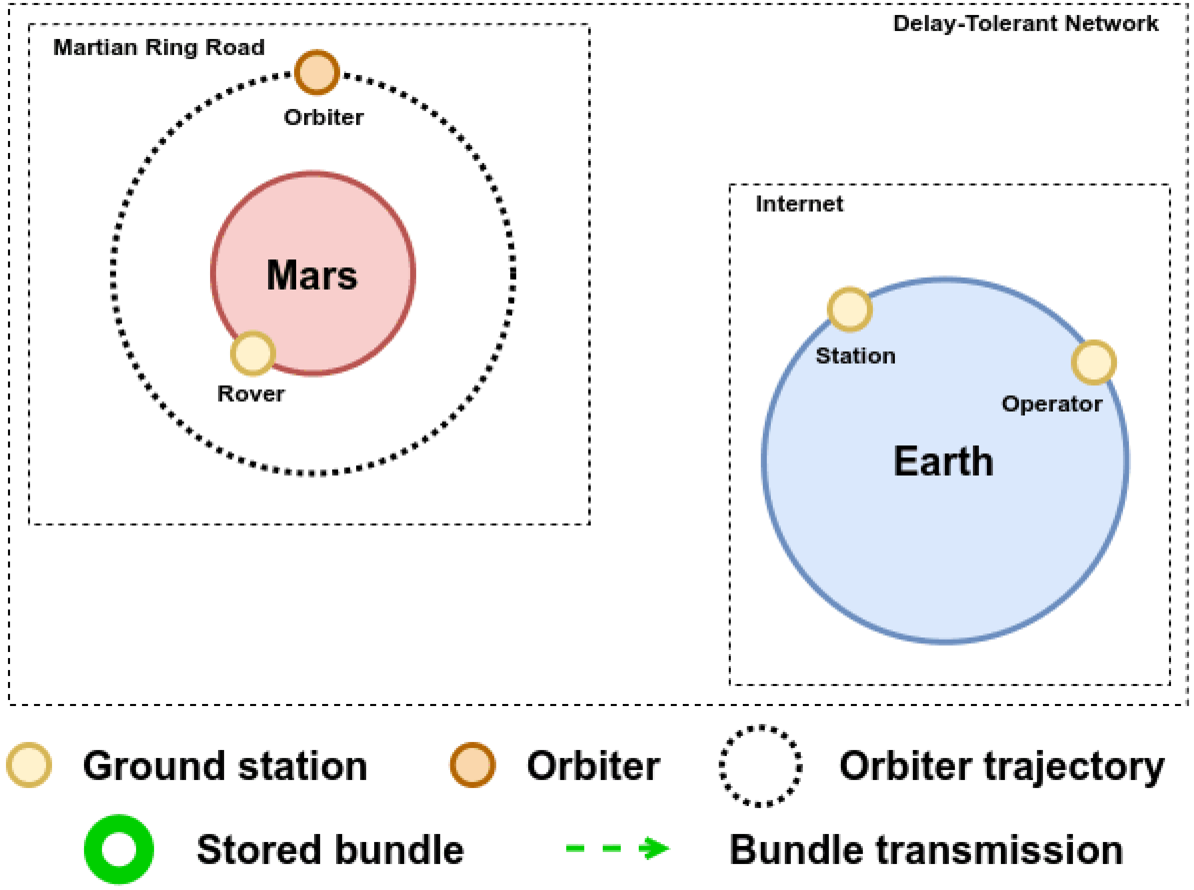
○ Ground station ● Orbiter ◌ Orbiter trajectory

○ Stored bundle ⇢ Bundle transmission

Bundle reaches Operator on Earth

Delay-Tolerant Networks and Bundle Protocol





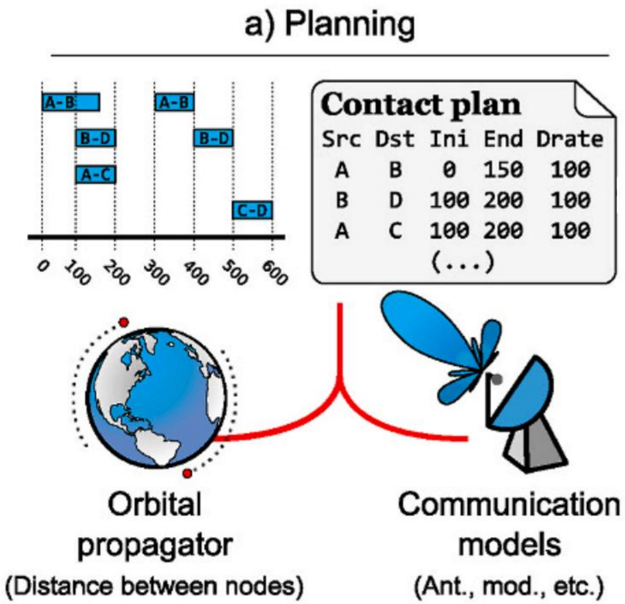
DTN operates as an overlay on diverse stacks,
bridging heterogeneous networks into a single, unified network.

Contact Graph Routing (CGR) framework & Schedule-Aware Bundle Routing (SABR)

SABR → specification CCSDS 734.3-B-1 [3];

CGR → an implementation (model and algorithmics) of the Jet Propulsion Laboratory [4].

a) CGR operates on a time varying graph: contact plan.



Contact plan

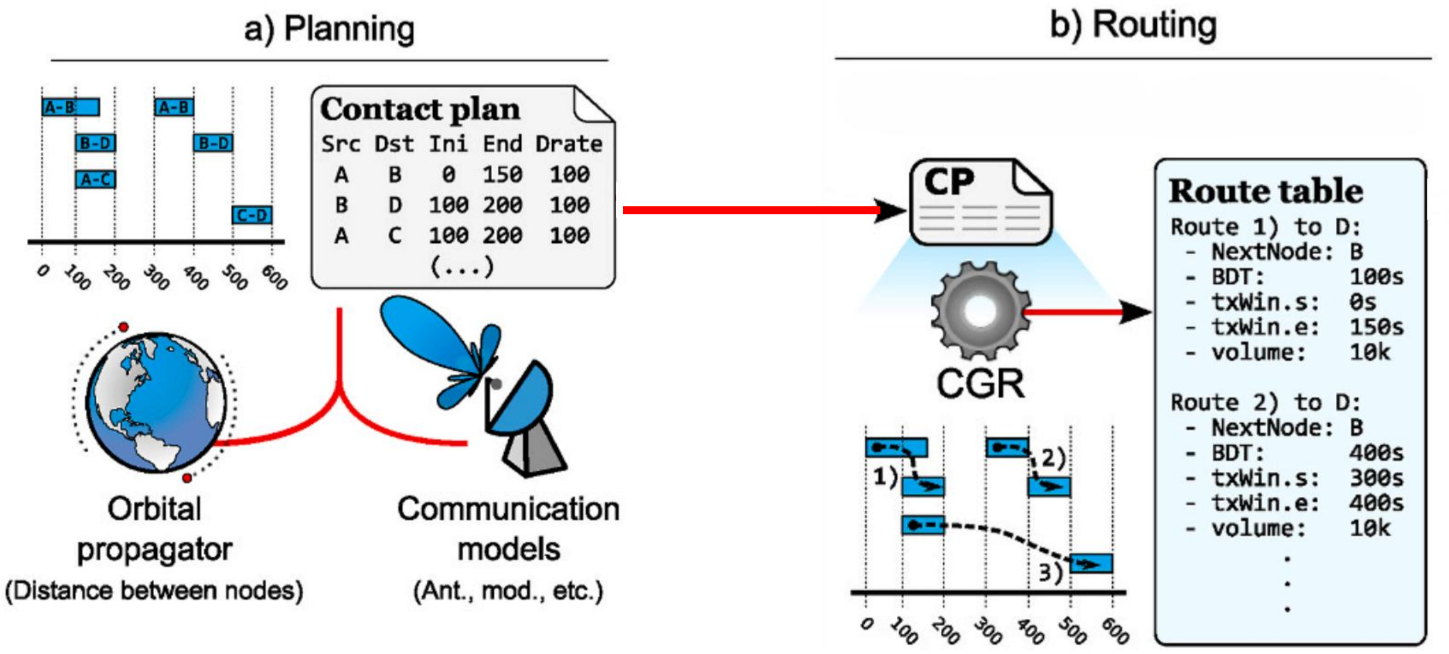
Src	Dst	Ini	End	Drate
A	B	0	150	100
B	D	100	200	100
A	C	100	200	100
(...)				

Pre/post processing of contact plan
(before the mission)

Contact Graph Routing (CGR) framework & Schedule-Aware Bundle Routing (SABR)

a) CGR operates on a time varying graph: contact plan.

b) CGR is based on Dijkstra and Yen's algorithms.



Contact plan

Src	Dst	Ini	End	Drate
A	B	0	150	100
B	D	100	200	100
A	C	100	200	100
(...)				

Route table

```
Route 1) to D:
 - NextNode: B
 - BDT:      100s
 - txWin.s:  0s
 - txWin.e:  150s
 - volume:   10k

Route 2) to D:
 - NextNode: B
 - BDT:      400s
 - txWin.s:  300s
 - txWin.e:  400s
 - volume:   10k
      .
      .
      .
```

Pre/post processing of contact plan (before the mission)

Path computation (complex algorithm)

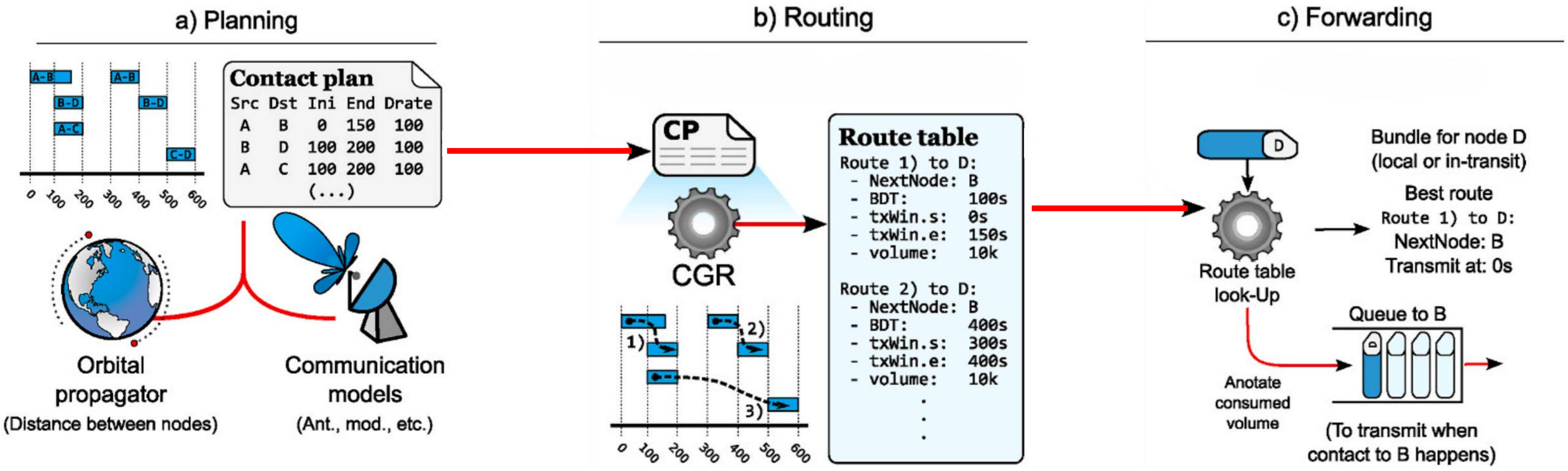
Contact Graph Routing (CGR) framework & Schedule-Aware Bundle Routing (SABR)

a) CGR operates on a time varying graph: contact plan.

b) CGR is based on Dijkstra and Yen's algorithms.

c) Existing routes are reused opportunistically.

Routing and Forwarding are intertwined



Contact plan

Src	Dst	Ini	End	Drate
A	B	0	150	100
B	D	100	200	100
A	C	100	200	100
(...)				

Route table

```
Route 1) to D:
 - NextNode: B
 - BDT:      100s
 - txWin.s:  0s
 - txWin.e:  150s
 - volume:   10k

Route 2) to D:
 - NextNode: B
 - BDT:      400s
 - txWin.s:  300s
 - txWin.e:  400s
 - volume:   10k
     .
     .
     .
```

Best route
Route 1) to D:
NextNode: B
Transmit at: 0s

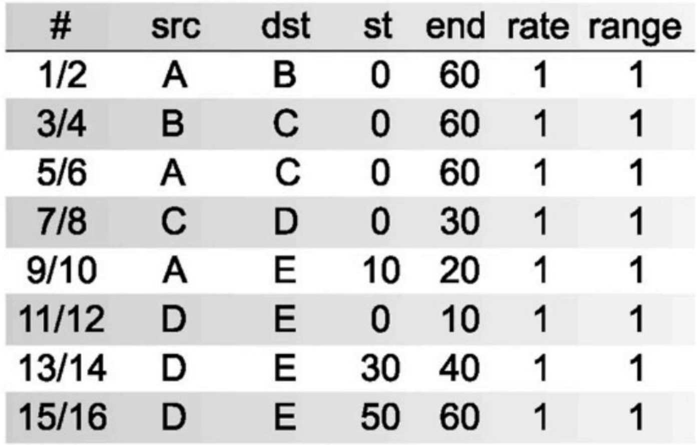
Pre/post processing of contact plan (before the mission)

Path computation (complex algorithm)

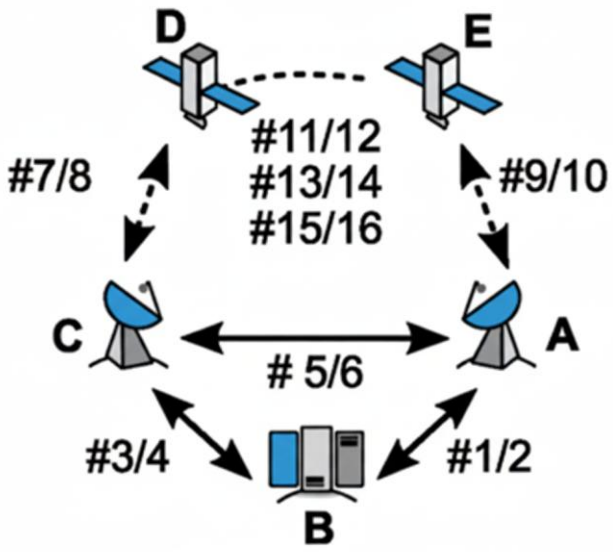
Path selection and queuing

Contact Graph: central data structure in SABR

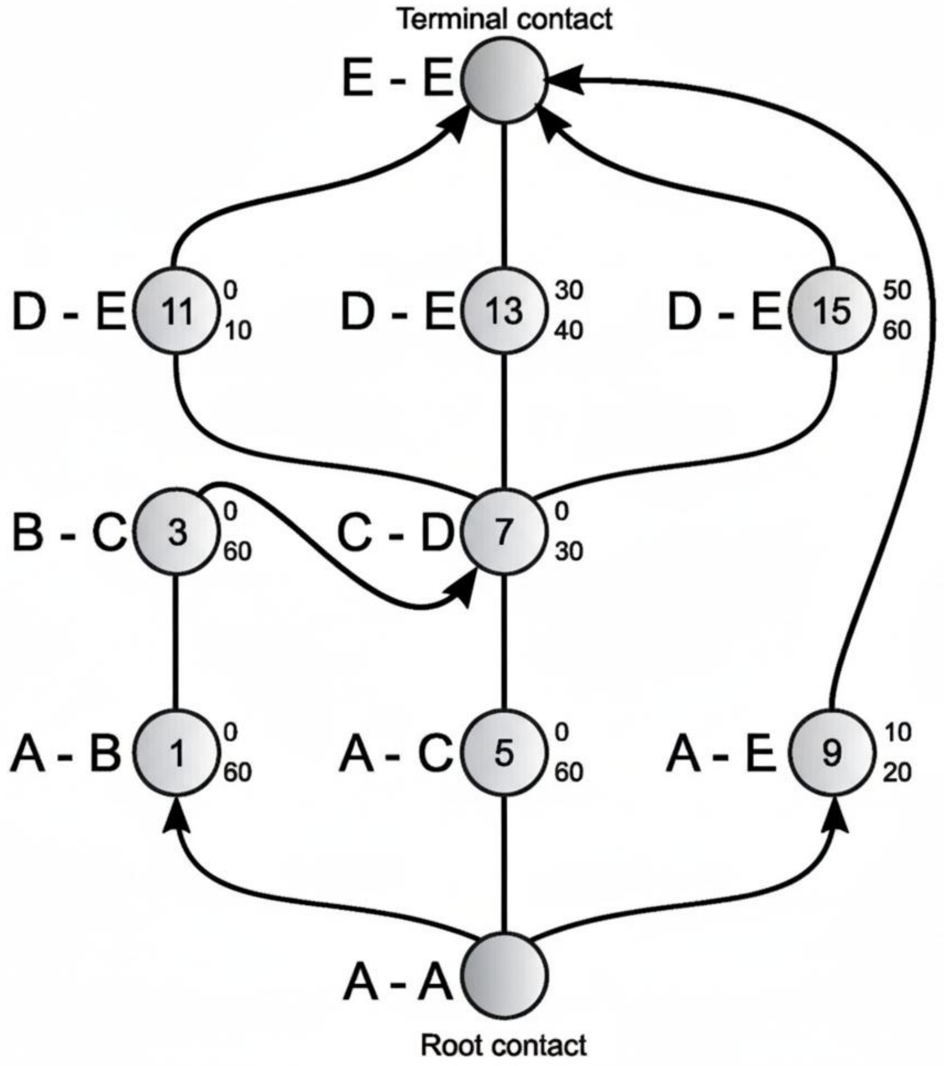
Contact plan table



#	src	dst	st	end	rate	range
1/2	A	B	0	60	1	1
3/4	B	C	0	60	1	1
5/6	A	C	0	60	1	1
7/8	C	D	0	30	1	1
9/10	A	E	10	20	1	1
11/12	D	E	0	10	1	1
13/14	D	E	30	40	1	1
15/16	D	E	50	60	1	1



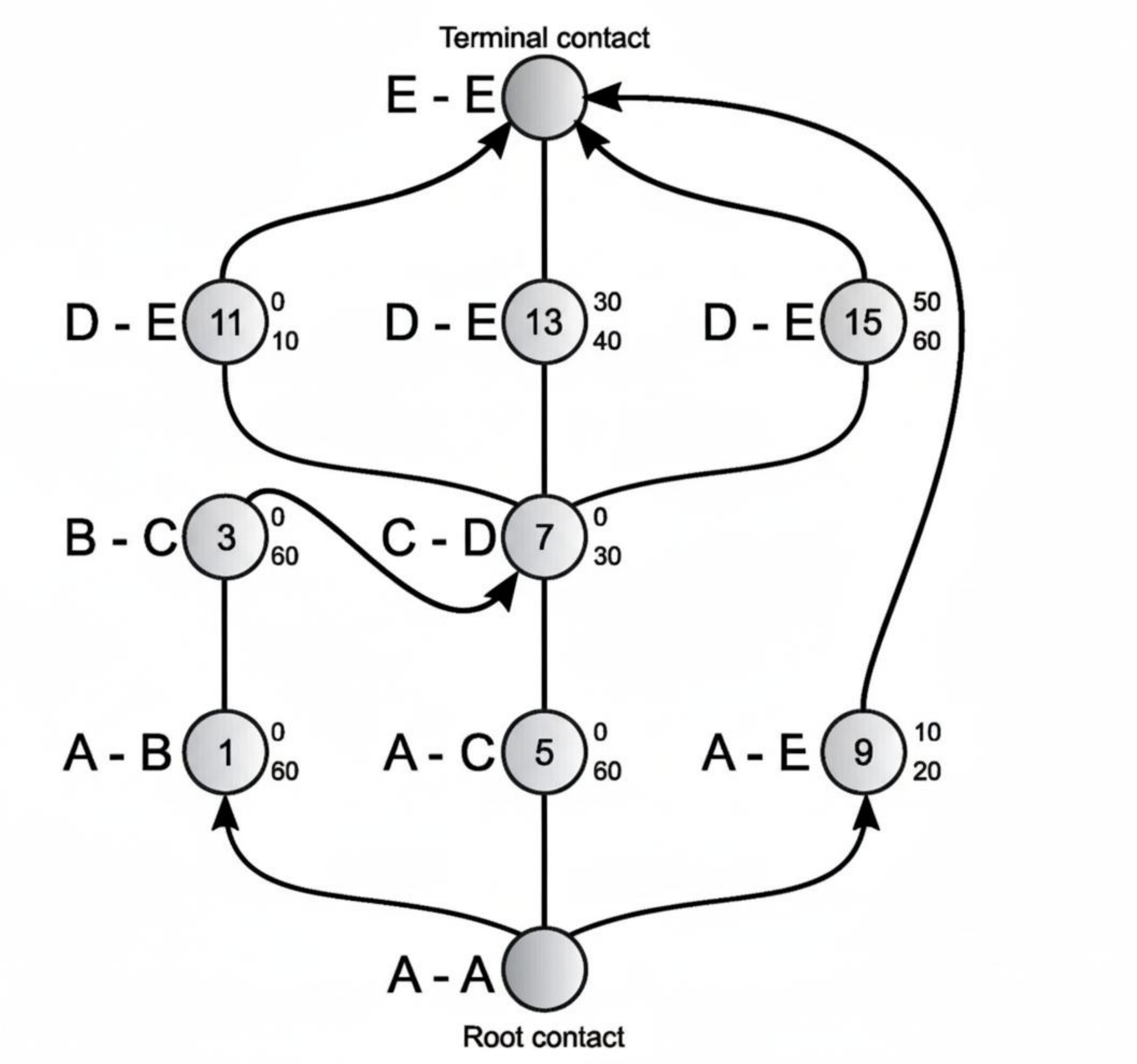
Static graph of the topology



Contact Graph

Contact Graph: central data structure in SABR

CGR can be represented as pathfinding on a contact graph where vertices are contacts and edges are potential periods of retention.



2nd Part

SABR/CGR and challenges

SABR/CGR and challenges

- CGR represents 15 years of collaborative efforts and remains a hot topic due to its complexity and scalability issues. Roy Gladden, chief of the Mars Relay Network, awaits DTN and described CGR as one of the three main subjects that need advancement.

- It is hard to implement new flavors in real production for benchmarking through emulation.

SABR/CGR and challenges

- CGR represents 15 years of collaborative efforts and remains a hot topic due to its complexity and scalability issues. Roy Gladden, chief of the Mars Relay Network, awaits DTN and described CGR as one of the three main subjects that need advancement.
- It is hard to implement new flavors in real production for benchmarking through emulation.
- Using prototyping languages for benchmarking through simulation is easier but infers the quality of the results.
- There is no modularity capabilities to ease research activities.

3rd Part

A-SABR Library

Contribution of A-SABR

A-SABR: (Adaptive Library for Schedule-Aware Bundle Routing)

modular, memory-safe, high-performance, compile-time configurable.

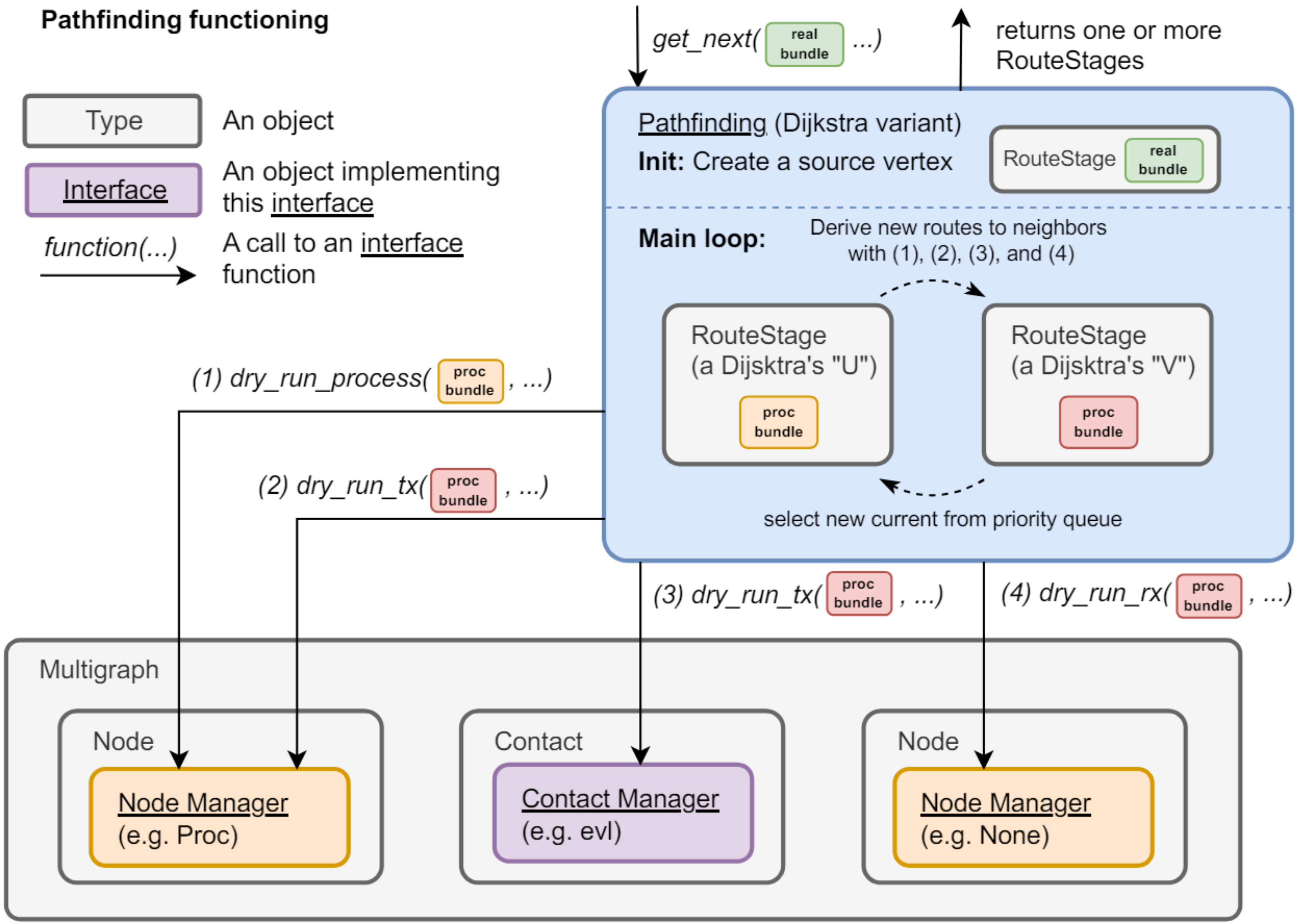
- Contact plan format: 3 formats (A-SABR native, ION, dtn-tvg-util)
- Node resource Management: 1 example
- Contact resource Management: 10 approaches
- Contact/node graph behavior is resolved at pathfinding by appropriate chaining/parenting
- Pathfinding: up to 12 approaches:
  - 6 approaches: ContactParentingPath/NodeParentingPath/HybridParentingPath
    ContactParentingTree/NodeParentingTree/HybridParentingTree
  - 6 approaches: 2 Alternative pathfinding approach
    to use with the single destination pathfinding approach (Path) variants
- Routing Mainframes : 3 approaches (Cgr, VolCgr, Spsn) All algos use a multigraph
- Routing storage: 2 approaches (one configurable)
- Distance calculation: 2 approaches

Different pathfinding algorithms and strategies in A-SABR

Algorithm	Pathfinding design	Feature
SPSN	Shortest-path tree volume aware search (size, priority)	Reusable shortest-path tree, native unicast/multicast support
CGR	Not considering bundle metrics (alternative pathfinding is required)	Route selection, construct new routes when selection fails
VolCGR	Replacing the alternative pathfinding approach with volume (and priority) aware search	Replace Yen's algo (often low efficiency) [5]

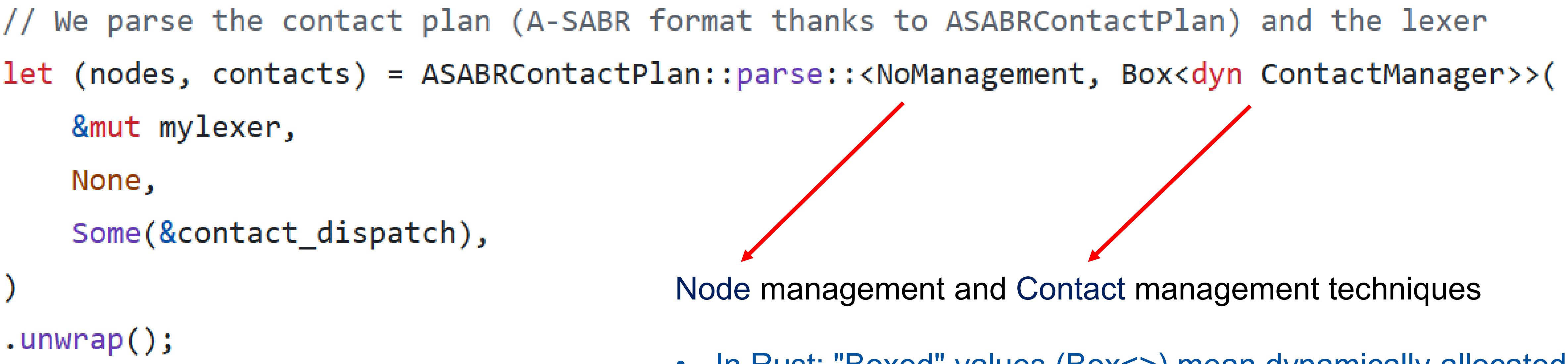
Alternative Pathfinding Strategy	Pathfinding design
FirstEnding	Suppress first ending contact of the last found route
FirstDepleted	Suppress the contact with the smallest original volume limit

Adaptive Library for Schedule-Aware Bundle Routing: Pathfinding workflow


Pathfinding functioning
Type
An object
Interface
An object implementing this interface
function(...)
A call to an interface function
get_next(real bundle ...)
returns one or more RouteStages
Pathfinding (Dijkstra variant)
Init: Create a source vertex
RouteStage real bundle
Main loop:
Derive new routes to neighbors with (1), (2), (3), and (4)
RouteStage (a Dijsktra's "U")
proc bundle
RouteStage (a Dijsktra's "V")
proc bundle
select new current from priority queue
(1) dry_run_process(proc bundle , ...)
(2) dry_run_tx(proc bundle , ...)
(3) dry_run_tx(proc bundle , ...)
(4) dry_run_rx(proc bundle , ...)
Multigraph
Node
Node Manager (e.g. Proc)
Contact
Contact Manager (e.g. evl)
Node
Node Manager (e.g. None)

[6]

Example 1: Basic usage (Parsing contact plan)



```
// We parse the contact plan (A-SABR format thanks to ASABRContactPlan) and the lexer
let (nodes, contacts) = ASABRContactPlan::parse::<NoManagement, Box<dyn ContactManager>>(
    &mut mylexer,
    None,
    Some(&contact_dispatch),
)
.unwrap();
```

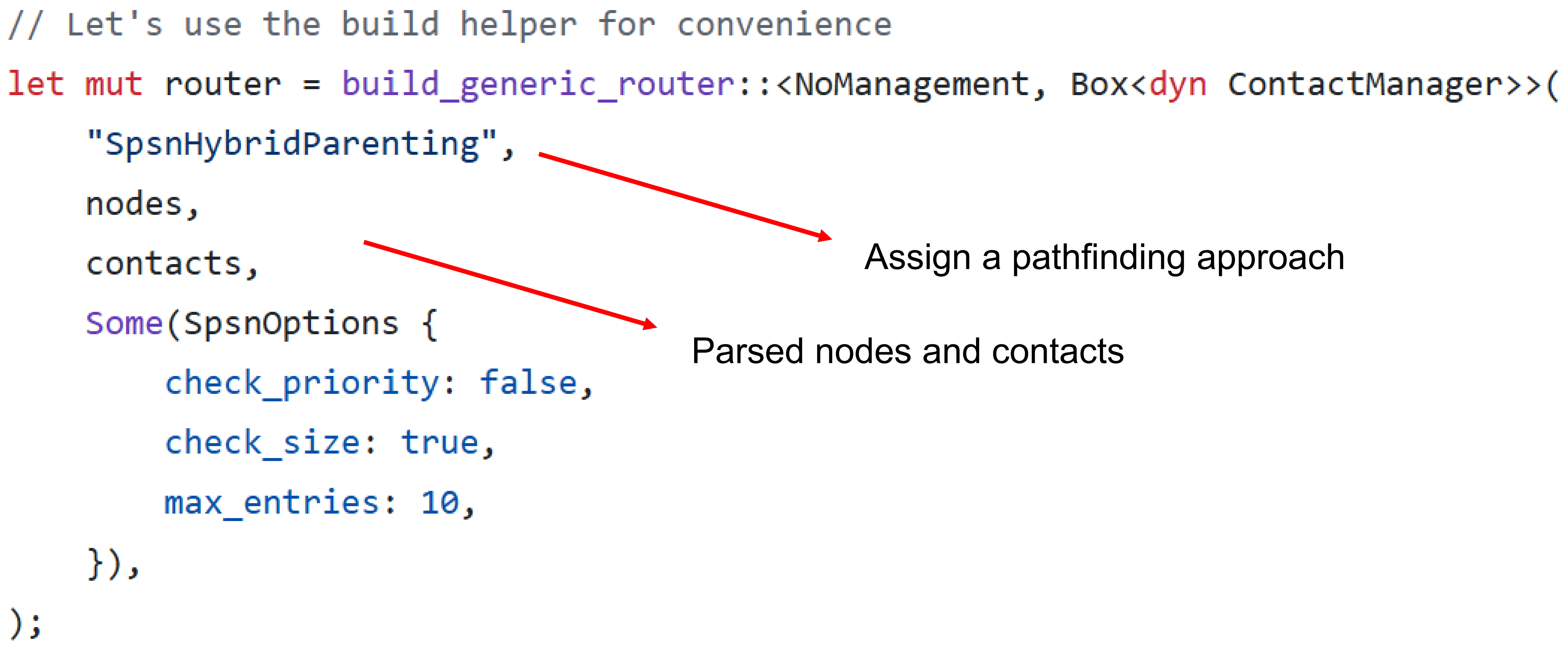
Node management and Contact management techniques

- In Rust: "Boxed" values (Box<>) mean dynamically allocated
- In Rust: dyn means that runtime polymorphism is involved
- ContactManager is an interface

We can assign different management techniques to individual contacts (reflected in the Contact Plan)

https://github.com/DTN-MTP/A-SABR/tree/main/examples/eto_management

Example 1: Basic usage (Building a router of one routing variant)



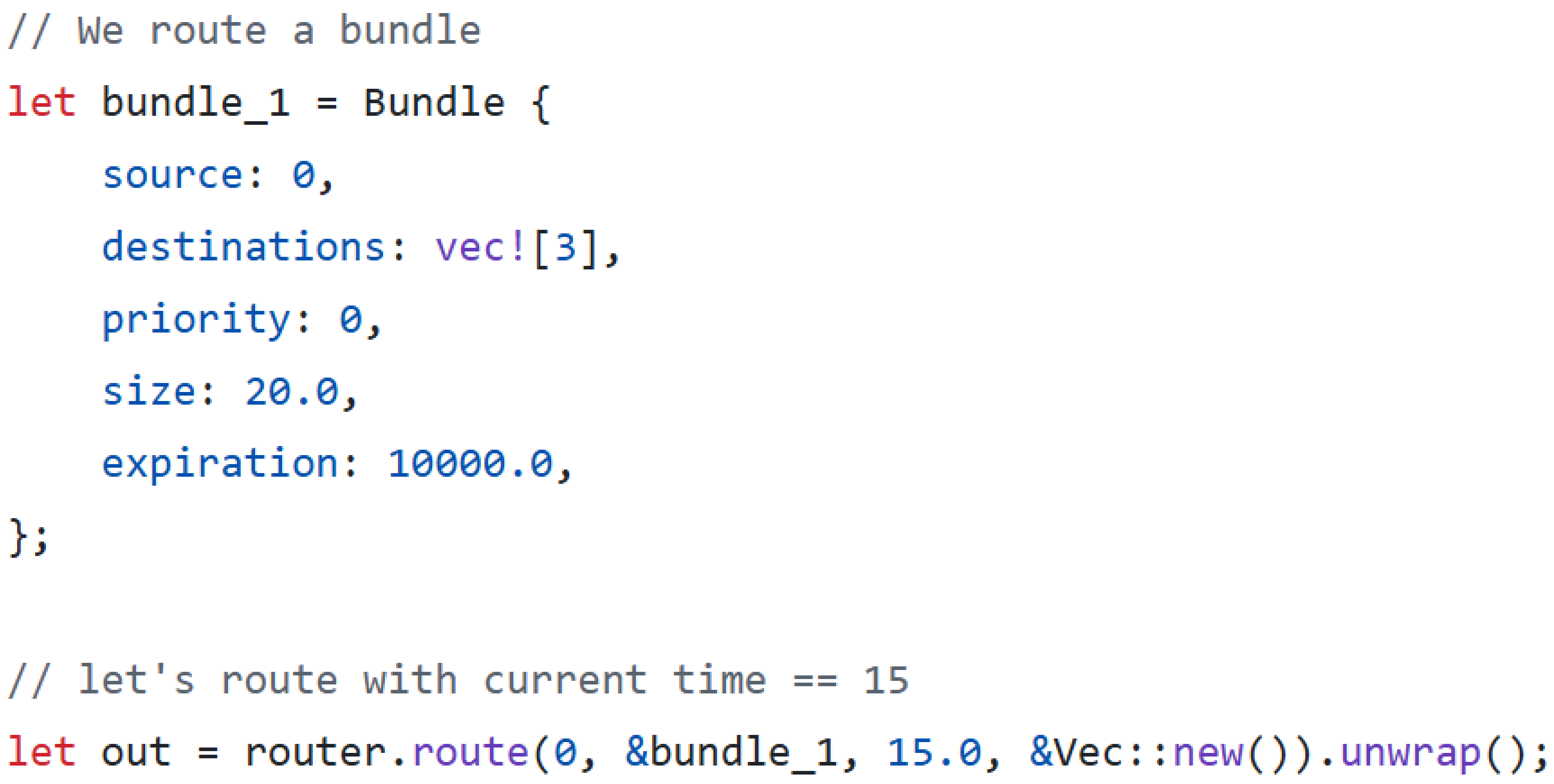
```
// Let's use the build helper for convenience
let mut router = build_generic_router::<NoManagement, Box<dyn ContactManager>>(
    "SpsnHybridParenting",
    nodes,
    contacts,
    Some(SpsnOptions {
        check_priority: false,
        check_size: true,
        max_entries: 10,
    }),
);
```

Assign a pathfinding approach

Parsed nodes and contacts

https://github.com/DTN-MTP/A-SABR/tree/main/examples/eto_management

Example 1: Basic usage (Routing a bundle)

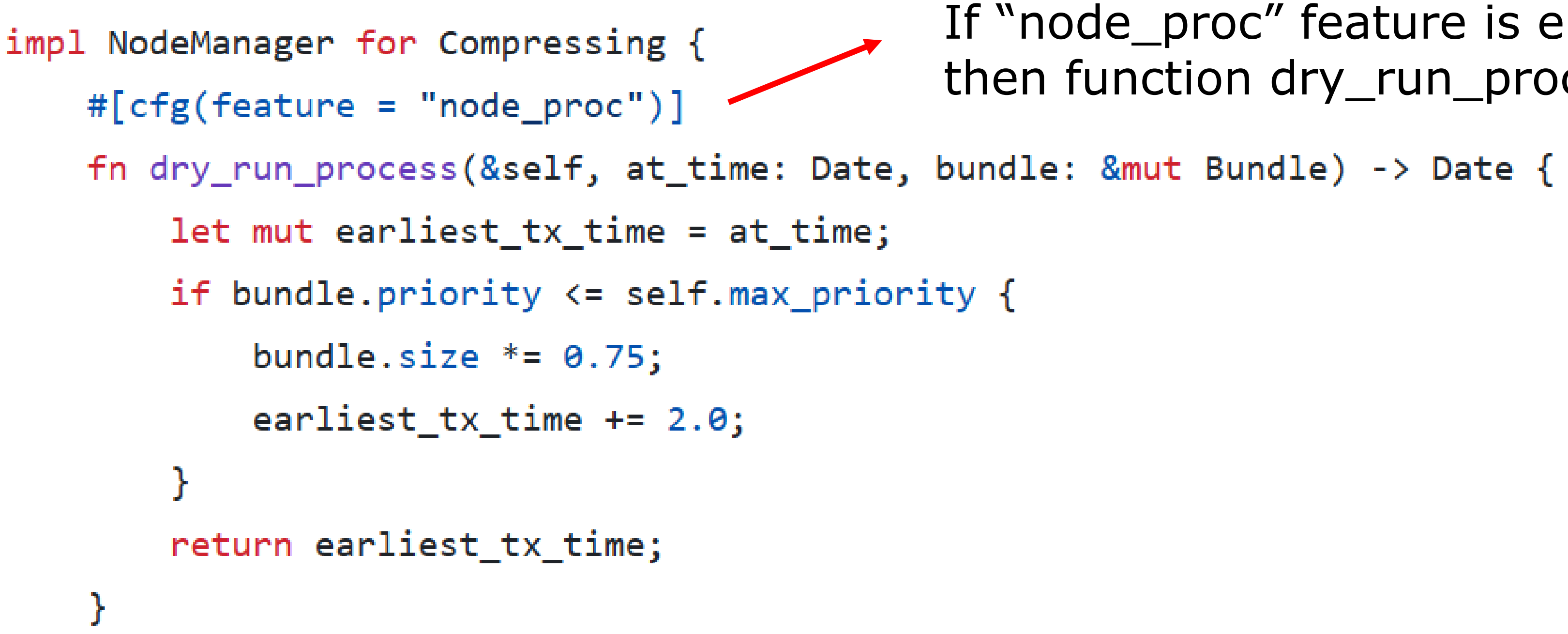


```rust
// We route a bundle
let bundle_1 = Bundle {
    source: 0,
    destinations: vec![3],
    priority: 0,
    size: 20.0,
    expiration: 10000.0,
};

// let's route with current time == 15
let out = router.route(0, &bundle_1, 15.0, &Vec::new()).unwrap();
```

https://github.com/DTN-MTP/A-SABR/tree/main/examples/eto_management

Example 2: Development (Customizing node manager for bundle compression)

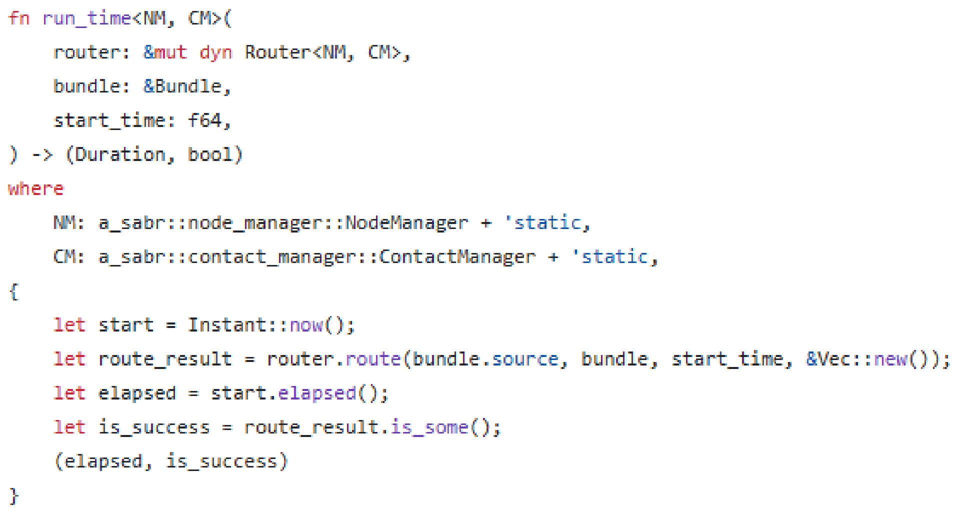


```
impl NodeManager for Compressing {
    #[cfg(feature = "node_proc")]
    fn dry_run_process(&self, at_time: Date, bundle: &mut Bundle) -> Date {
        let mut earliest_tx_time = at_time;
        if bundle.priority <= self.max_priority {
            bundle.size *= 0.75;
            earliest_tx_time += 2.0;
        }
        return earliest_tx_time;
    }
```

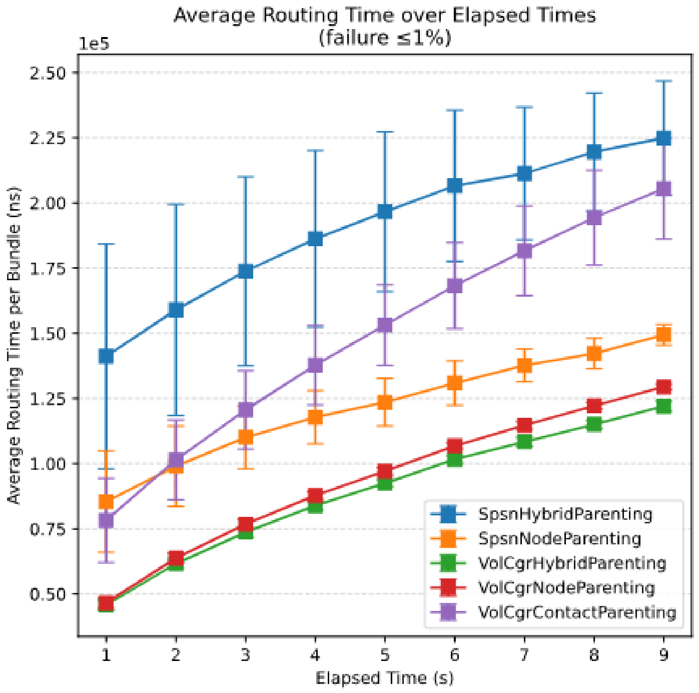
compile-time configurable:
If “node_proc” feature is enabled,
then function dry_run_process will be compiled.

https://github.com/DTN-MTP/A-SABR/tree/main/examples/bundle_processing

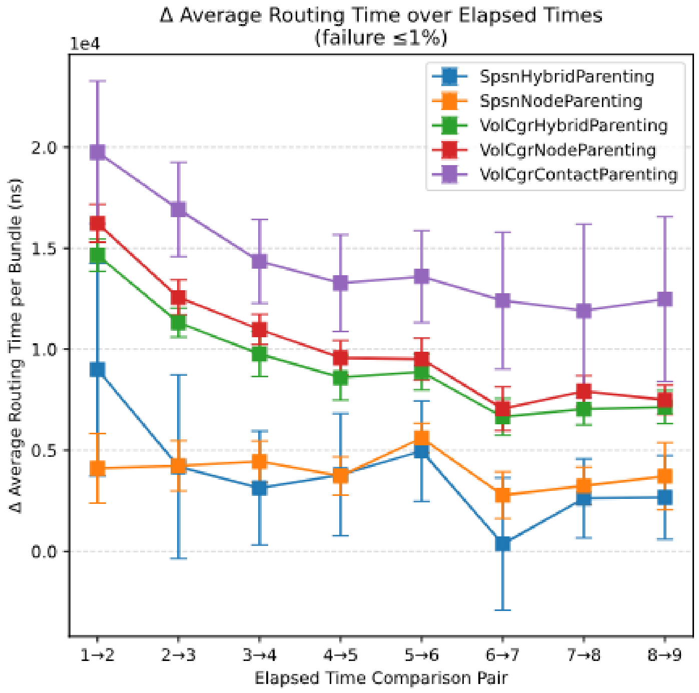
Example 3: Benchmark



```rust
fn run_time<NM, CM>(
    router: &mut dyn Router<NM, CM>,
    bundle: &Bundle,
    start_time: f64,
) -> (Duration, bool)
where
    NM: a_sabr::node_manager::NodeManager + 'static,
    CM: a_sabr::contact_manager::ContactManager + 'static,
{
    let start = Instant::now();
    let route_result = router.route(bundle.source, bundle, start_time, &Vec::new());
    let elapsed = start.elapsed();
    let is_success = route_result.is_some();
    (elapsed, is_success)
}
```



(a) Grid search on elapsed time



(b) Comparison of 2 adjacent elapsed times

https://github.com/DTN-MTP/A-SABR-processing-benchmark

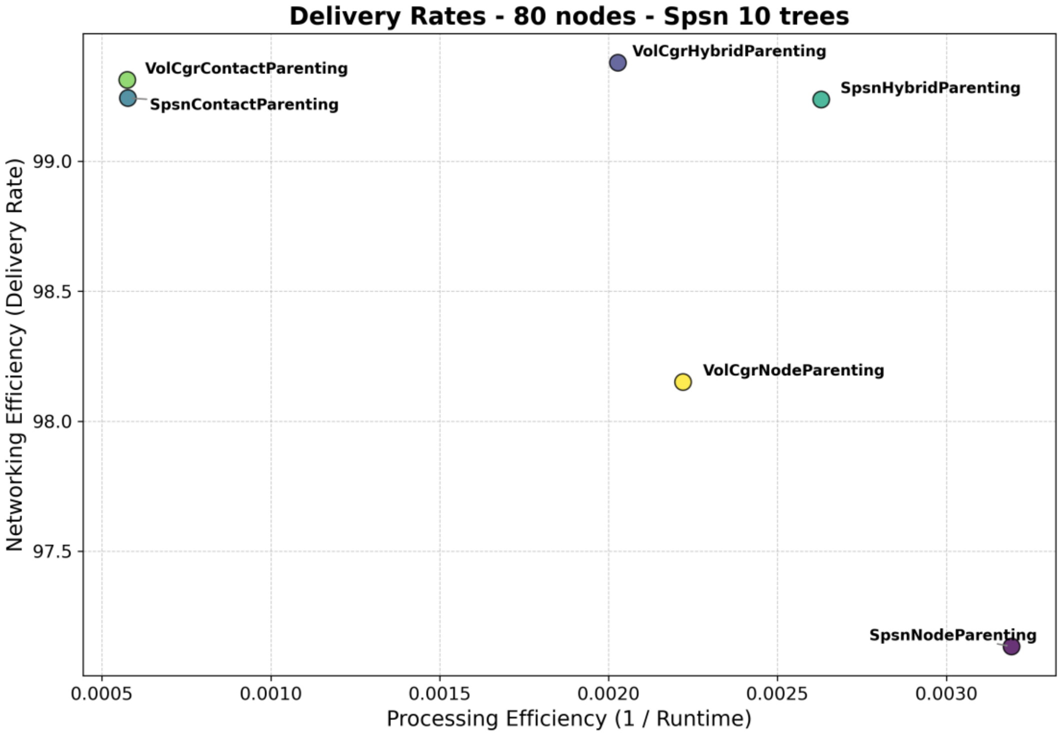
4th

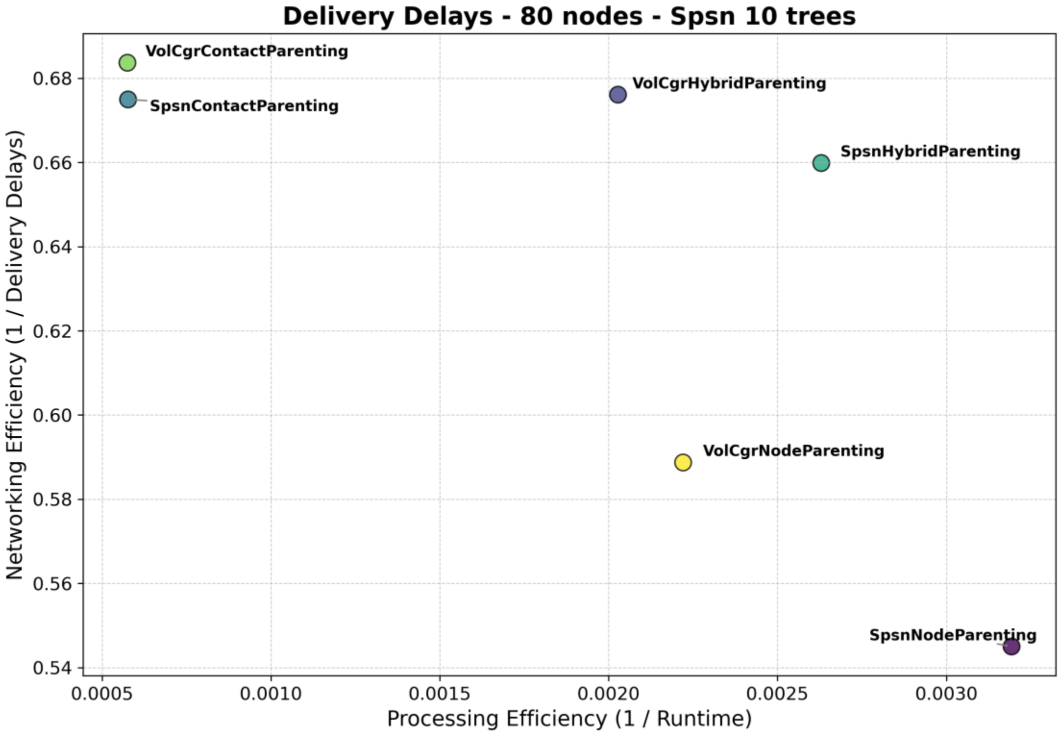
Evaluation

Part

Networking performance (“online” evaluation) on ring road scenario

- Best-performing algorithm: VolCgr.
- Spsn remains configurable to lower the processing pressure. (on small contact graph)





Delivery Rate (y axe: higher the better)

vs.

Processing time (x axe: higher the better)

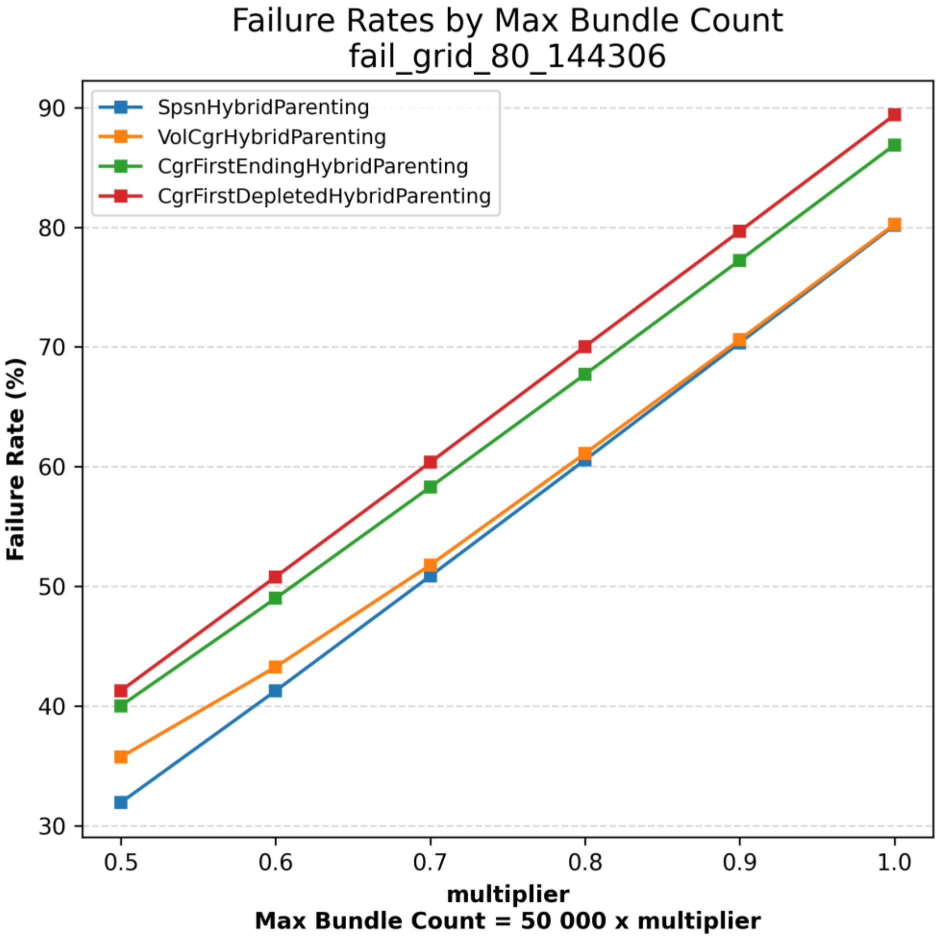
Delivery Delay (y axe: higher the better)

vs.

Processing time (x axe: higher the better)

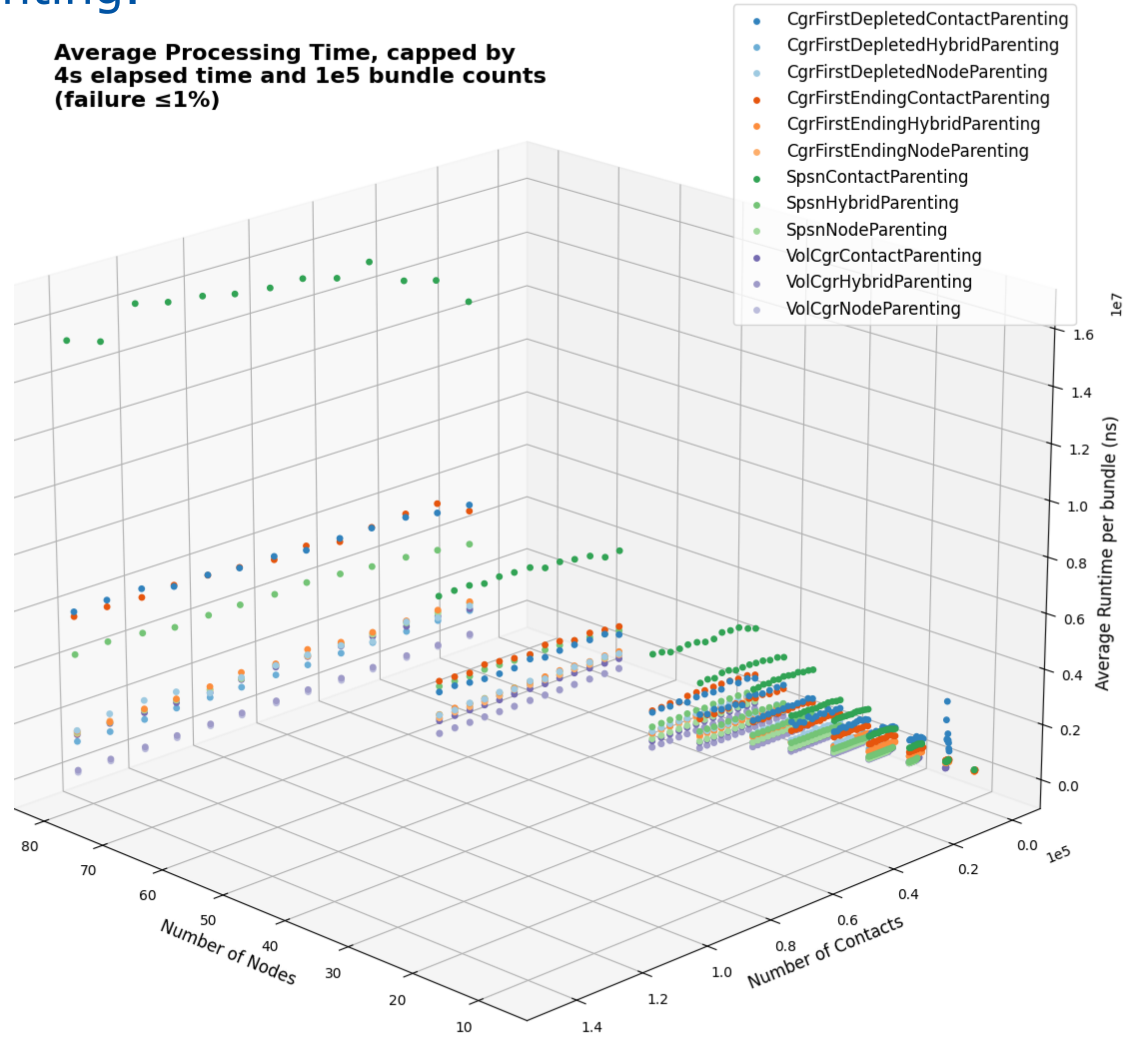
Computational performance (“off-line”) benchmarking

- Best-performing algorithm: VolCgr.
- High computational pressure of Contact Parenting.



Routing failure rate (y axe: lower the better) vs. bundle count (with congestion)

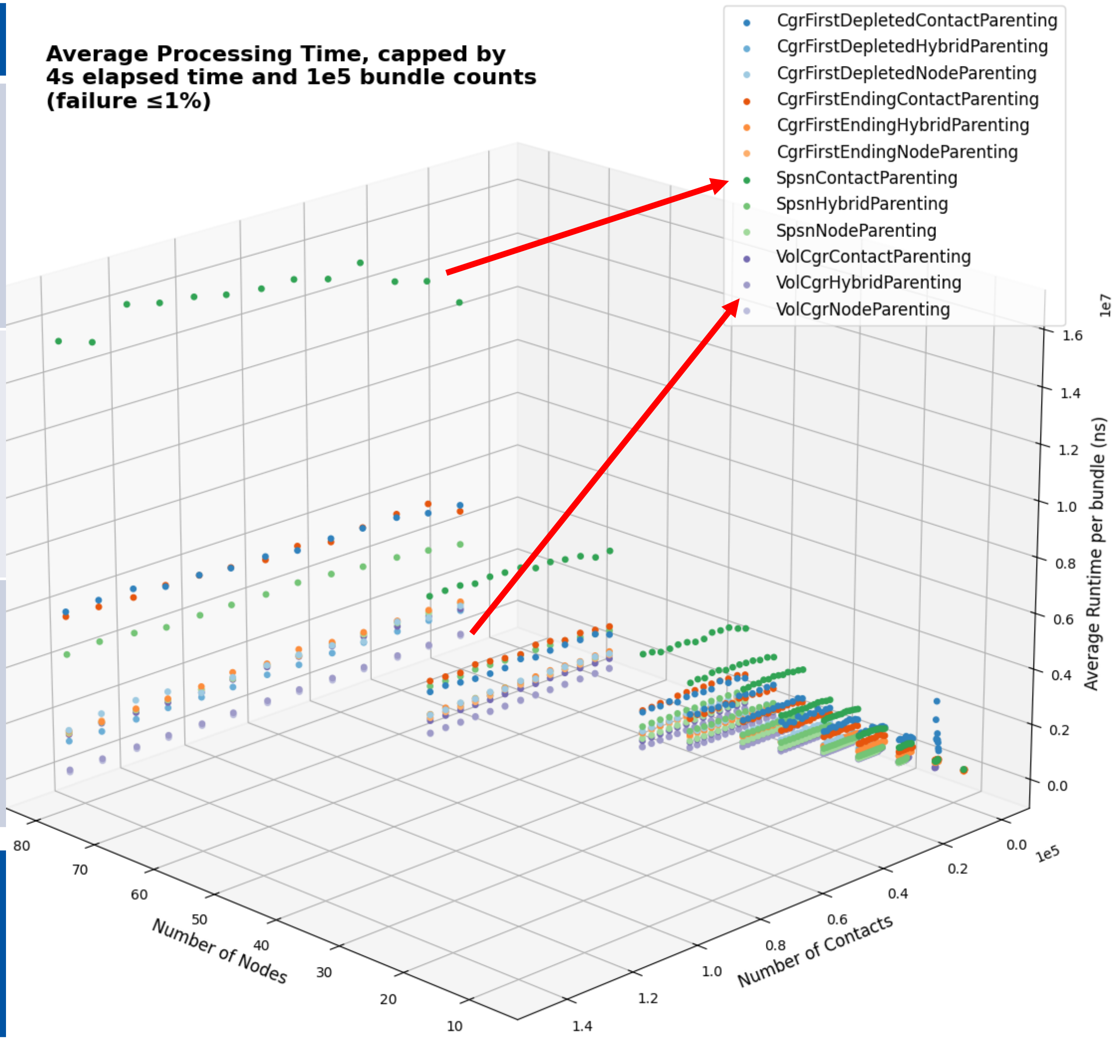
(Transportation scenario, 80 nodes, 144306 contacts)



Bundle scheduling processing time benchmarking (z axe: lower the better)

Computational performance (“off-line”) benchmarking

Algo	Pathfinding design	Feature
SPSN	Shortest-path tree volume aware search (size, priority)	Reusable shortest-path tree, native unicast/multicast support
CGR	Not considering bundle metrics (alternative pathfinding is required)	Route selection, construct new routes when selection fails
VolCGR	Replacing the alternative pathfinding approach with volume (and priority) aware search	Replace Yen’s algo (often low efficiency)

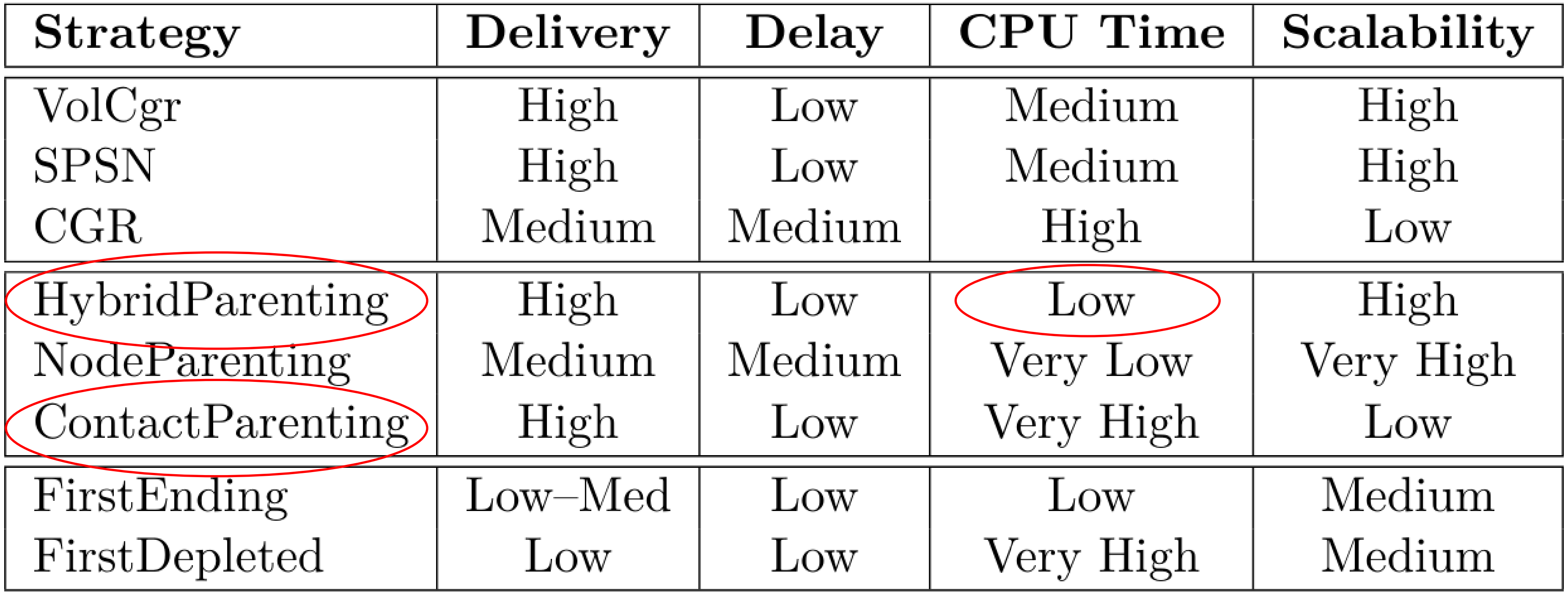


Alternative Pathfinding Strategy	Pathfinding design
FirstEnding	Suppress first ending contact of the last found route
FirstDepleted	Suppress the contact with the smallest original volume limit

5th Part

Conclusion and Future works

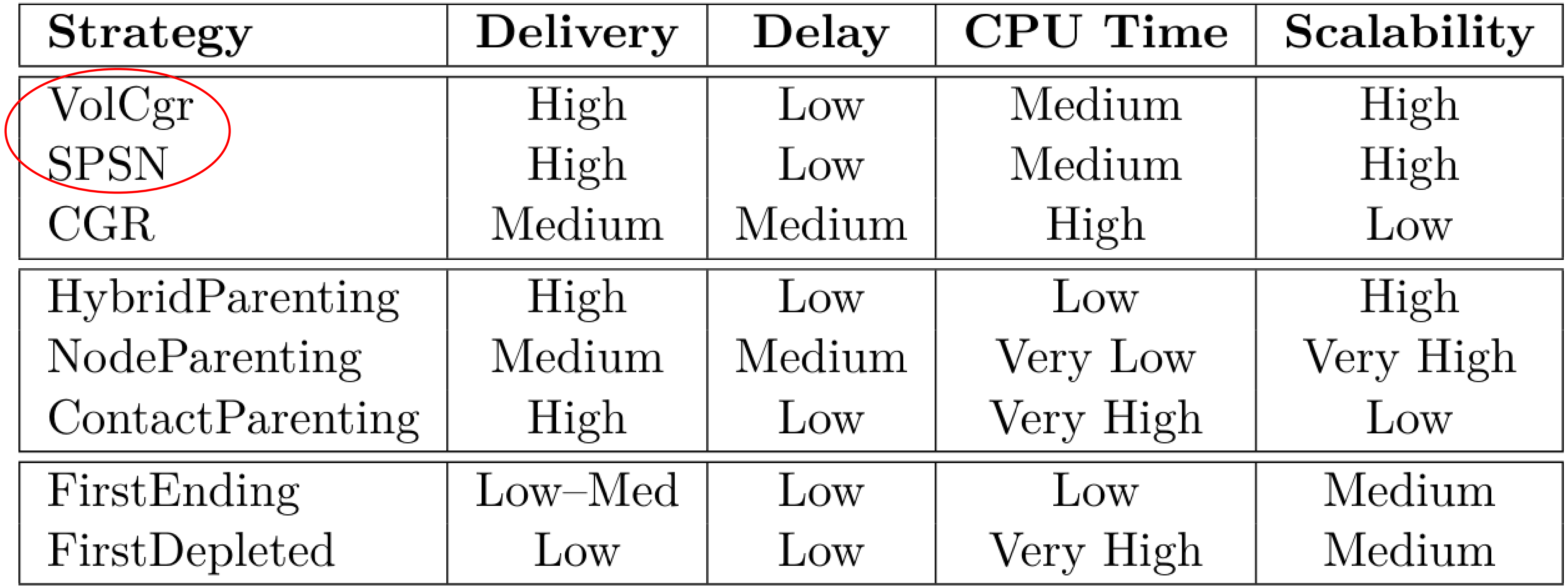
Tradeoffs Across Routing mainframes, Pathfinding, composite strategies



Strategy	Delivery	Delay	CPU Time	Scalability
VolCgr	High	Low	Medium	High
SPSN	High	Low	Medium	High
CGR	Medium	Medium	High	Low
HybridParenting	High	Low	Low	High
NodeParenting	Medium	Medium	Very Low	Very High
ContactParenting	High	Low	Very High	Low
FirstEnding	Low–Med	Low	Low	Medium
FirstDepleted	Low	Low	Very High	Medium

1. HybridParenting approach delivers networking performance comparable to ContactParenting while maintaining significantly lower computational costs.

Tradeoffs Across Routing mainframes, Pathfinding, composite strategies



Strategy	Delivery	Delay	CPU Time	Scalability
VolCgr	High	Low	Medium	High
SPSN	High	Low	Medium	High
CGR	Medium	Medium	High	Low
HybridParenting	High	Low	Low	High
NodeParenting	Medium	Medium	Very Low	Very High
ContactParenting	High	Low	Very High	Low
FirstEnding	Low–Med	Low	Low	Medium
FirstDepleted	Low	Low	Very High	Medium

2. VolCgr outperforms existing methods in our realistic simulation scenarios, SPSN remains most adapable for performance/features/cost tradeoffs.

Conclusion

1. HybridParenting approach delivers networking performance comparable to ContactParenting while maintaining significantly lower computational costs.

2. VolCgr outperforms existing methods in our realistic simulation scenarios, SPSN remains most adapable for performance/features/cost tradeoffs

3. A-SABR offers a platform that is both extensible and efficient.

Future work

1. Extending node-level management capabilities.
2. Integrating advanced scheduling strategies.
3. Supporting dynamic contact plan updates.
4. Enabling native interfacing with real-time DTN stacks.

Thank you!
Merci !

Sources

- *[1] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant networking architecture," Internet Requests for Comments, RFC Editor, RFC 4838, April 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4838.txt*
- *[2] K. Scott, S. Burleigh, and E. Birrane, "Bundle protocol version 7," Internet Requests for Comments, RFC Editor, RFC 9171, January 2022. [Online]. Available: http://www.rfc-editor.org/rfc/rfc9171.txt*
- *[3] Consultative Committee for Space Data Systems (CCSDS), "Schedule-Aware Bundle Routing (blue book, recommended standard CCSDS 734.3-B-1," https://ccsds.org/Pubs/734x3b1.pdf, July 2019.*
- *[4] Fraire, Juan A., Olivier De Jonckère, and Scott C. Burleigh. "Routing in the space internet: A contact graph routing tutorial." Journal of Network and Computer Applications 174 (2021): 102884.*
- *[5] De Jonckère, Olivier, Juan A. Fraire, and Scott Burleigh. "On the tractability of yen's algorithm and contact graph modeling in contact graph routing." 2023 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE). IEEE, 2023.*
- *[6] Olivier De Jonckère, Longrui Ma, and Juan A. Fraire. A-sabr: The adaptive library for schedule-aware bundle routing. In Proceedings of the Space-Terrestrial Internetworking Workshop (STINT), co-located with IEEE WiSEE 2025, 2025. To appear; submitted via IEEE WiSEE submission portal in EDAS.*