TEGRA: Efficient Ad-Hoc Analytics on Evolving Graphs

Anand Iyer[*▲], *Qifan Pu*[◆], *Kishan Patel*[■], *Joseph E. Gonzalez*[▲], *Ion Stoica*[▲]

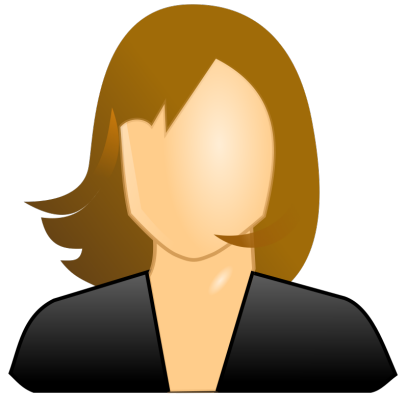
[*]Microsoft Research [▲]UC Berkeley [◆]Google [■]Two Sigma

USENIX NSDI, April 2021

Microsoft Research

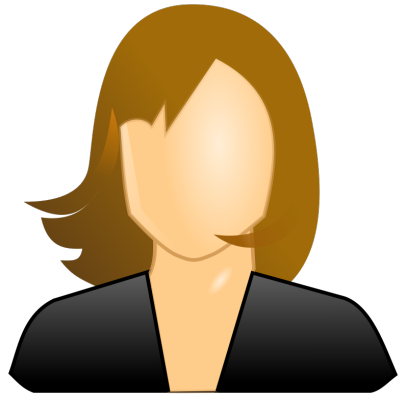
riselab UC Berkeley

Meet **Carol**



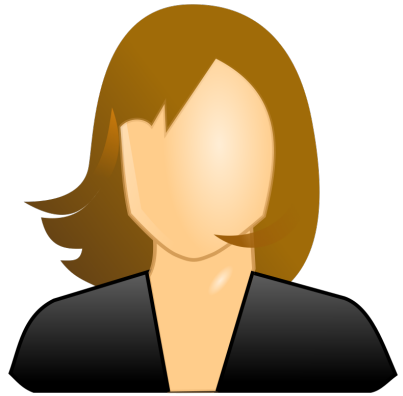
Network Administrator
ACME Cellular Company

Meet **Carol**



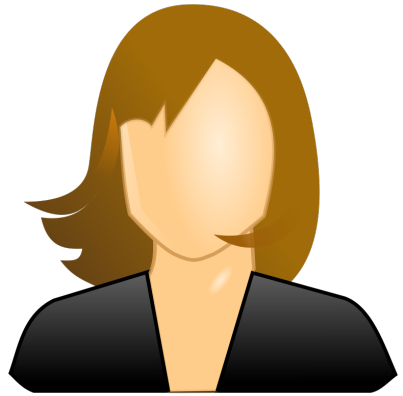
Network Administrator
ACME Cellular Company

Meet **Carol**



Network Administrator
ACME Cellular Company

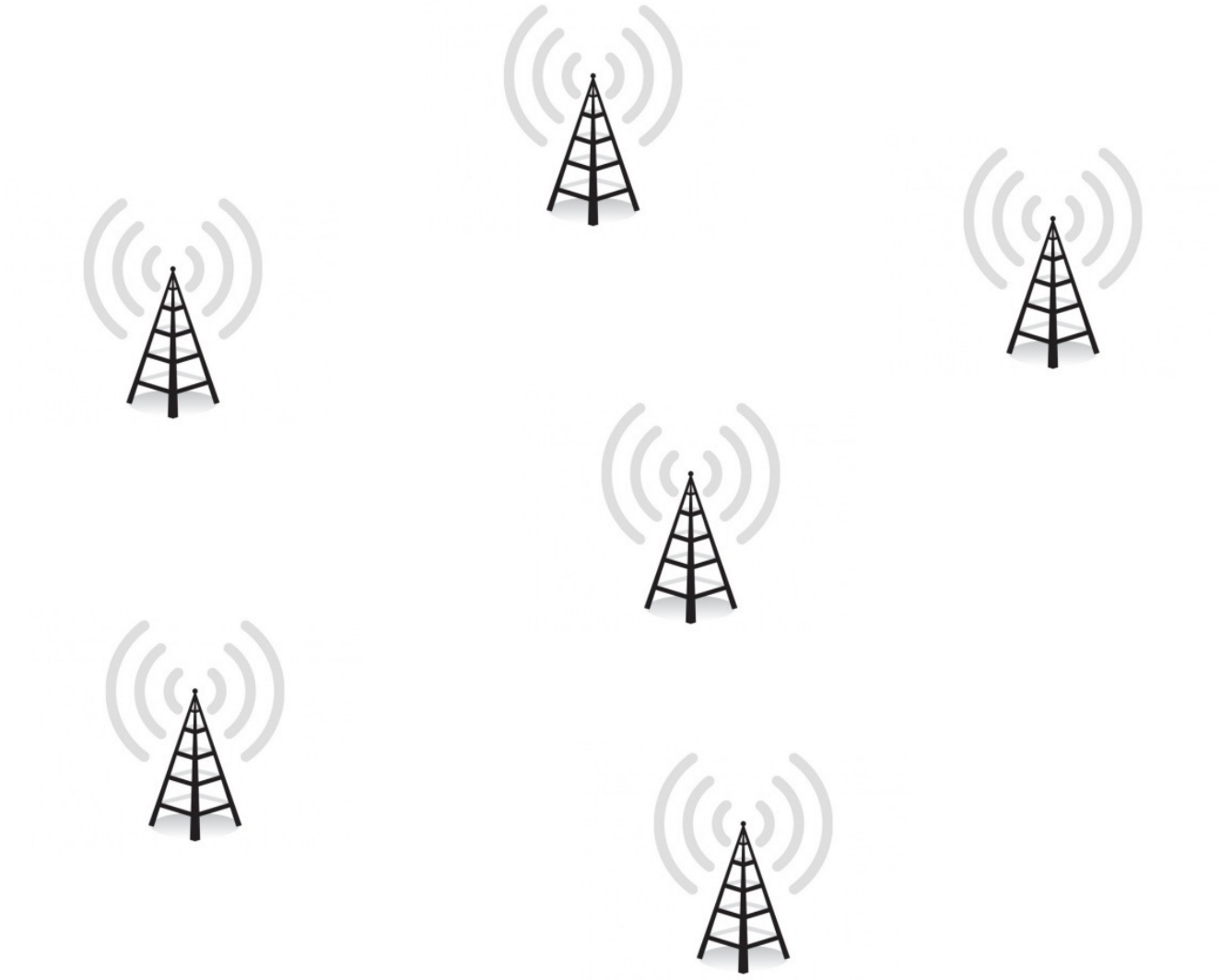
Meet **Carol**



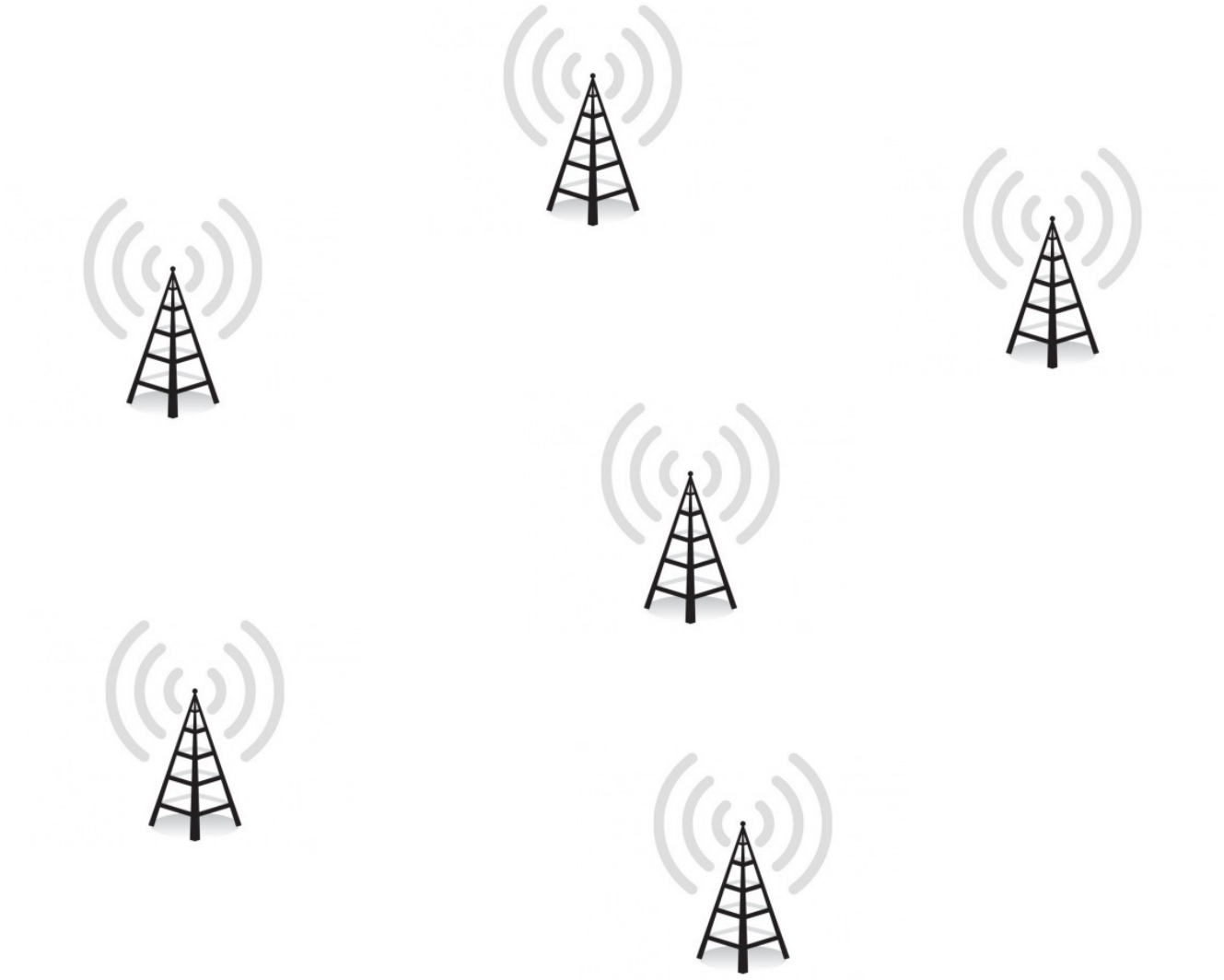
Network Administrator
ACME Cellular Company

What is the **reason** for poor download speeds for many users at 9am?

How **Carol** does this

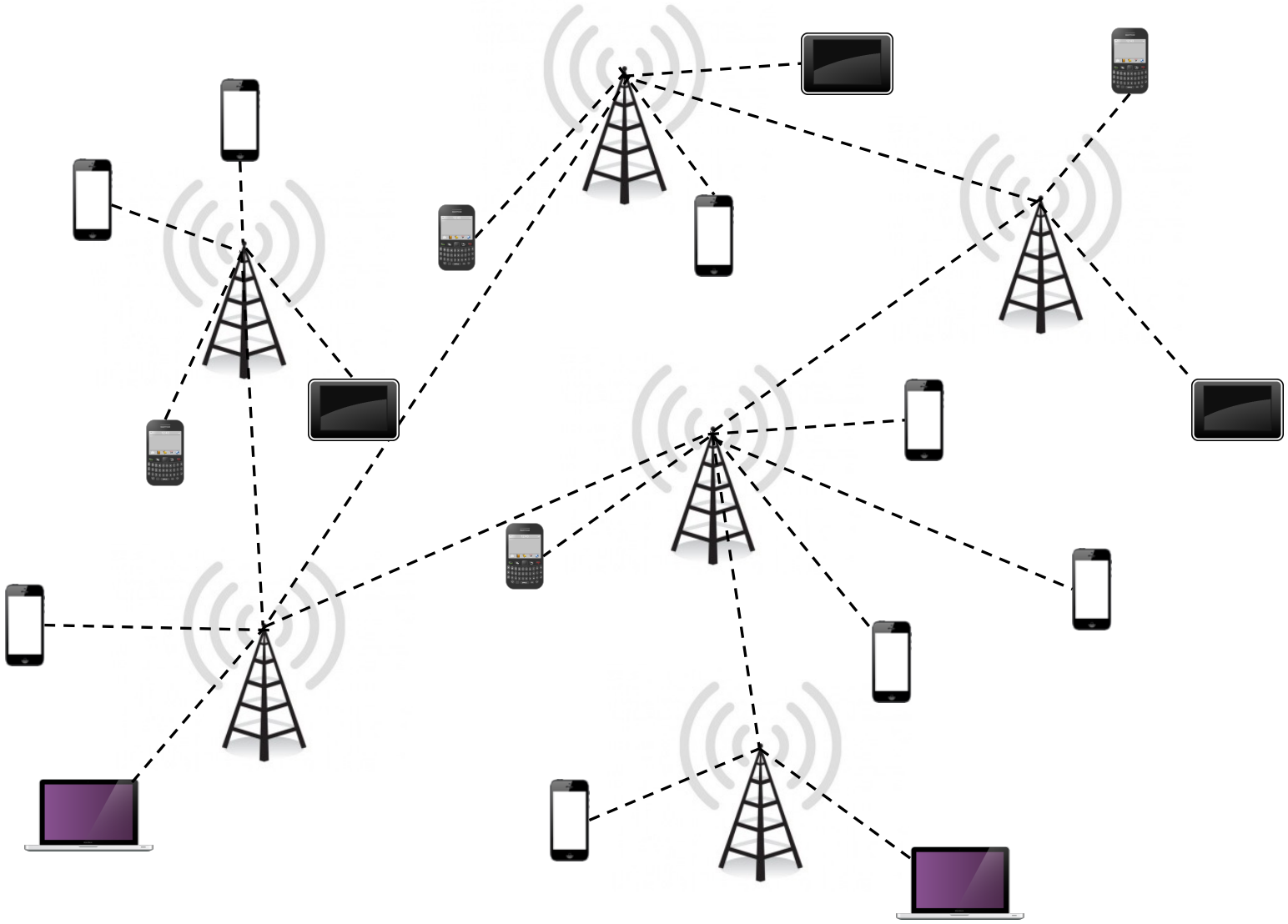


How **Carol** does this



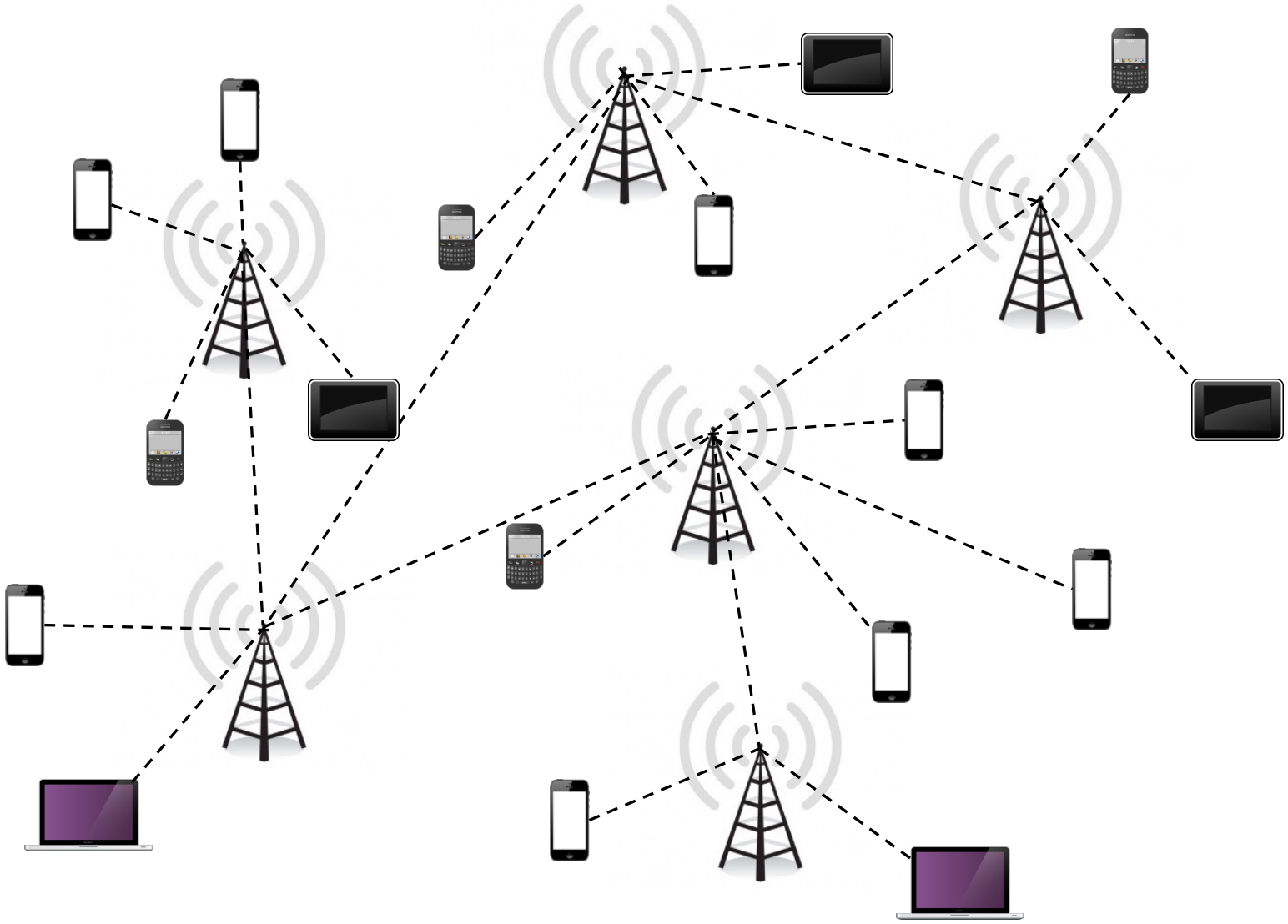
What did the network look like at 9am?

How **Carol** does this



What did the network look like at 9am?

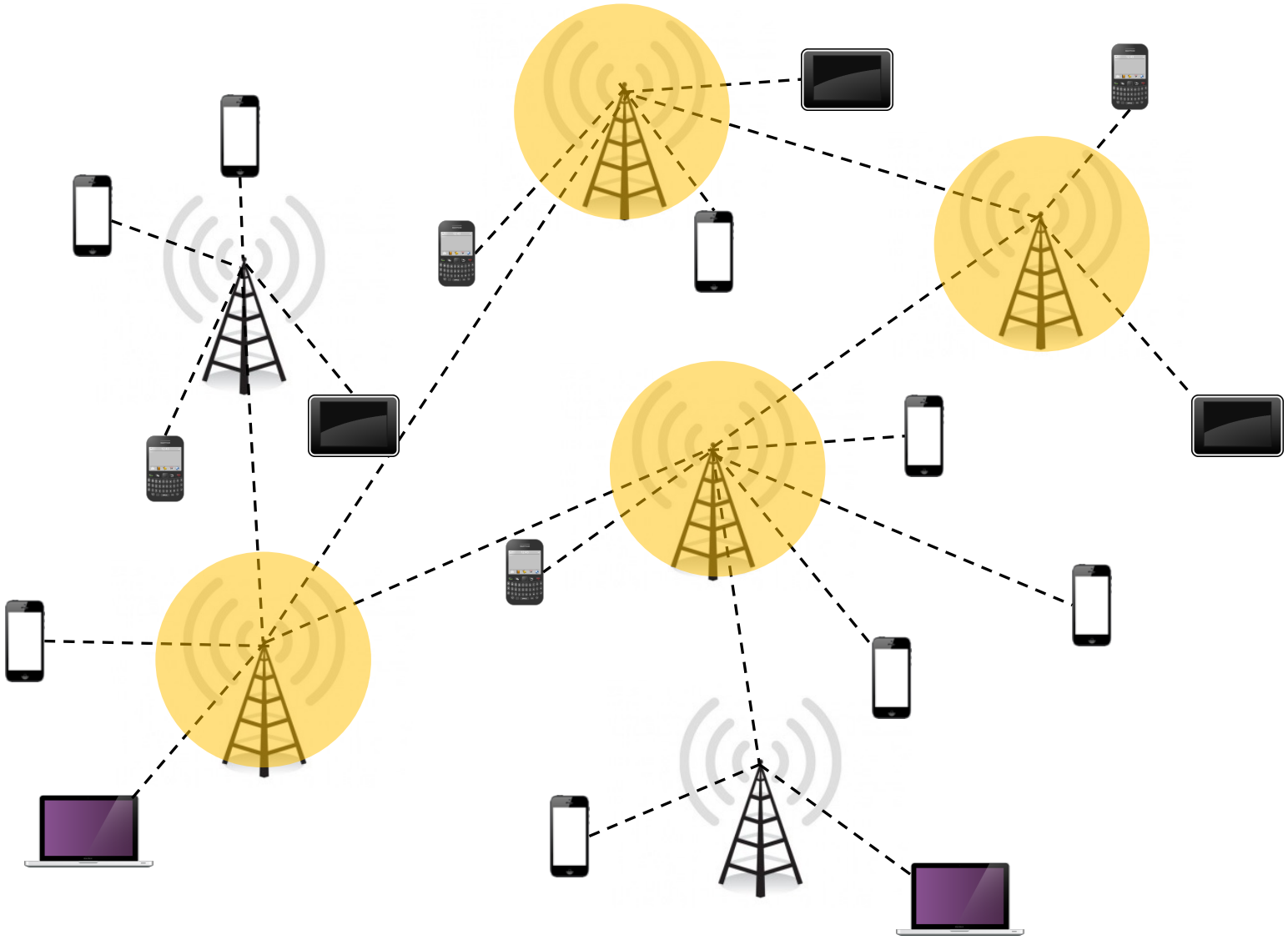
How **Carol** does this



What did the network look like at 9am?

Which towers were congested then?

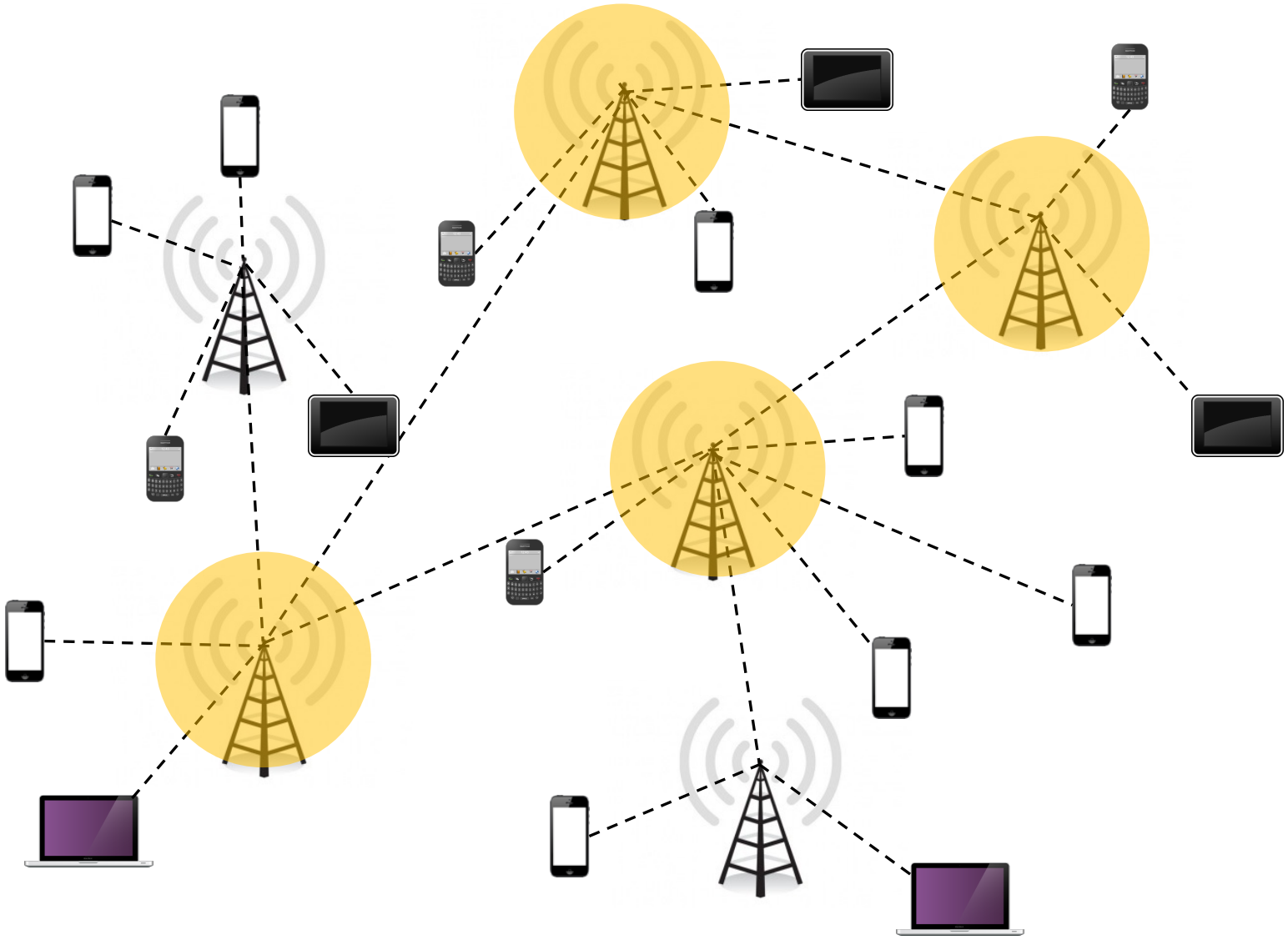
How **Carol** does this



What did the network look like at 9am?

Which towers were congested then?

How **Carol** does this

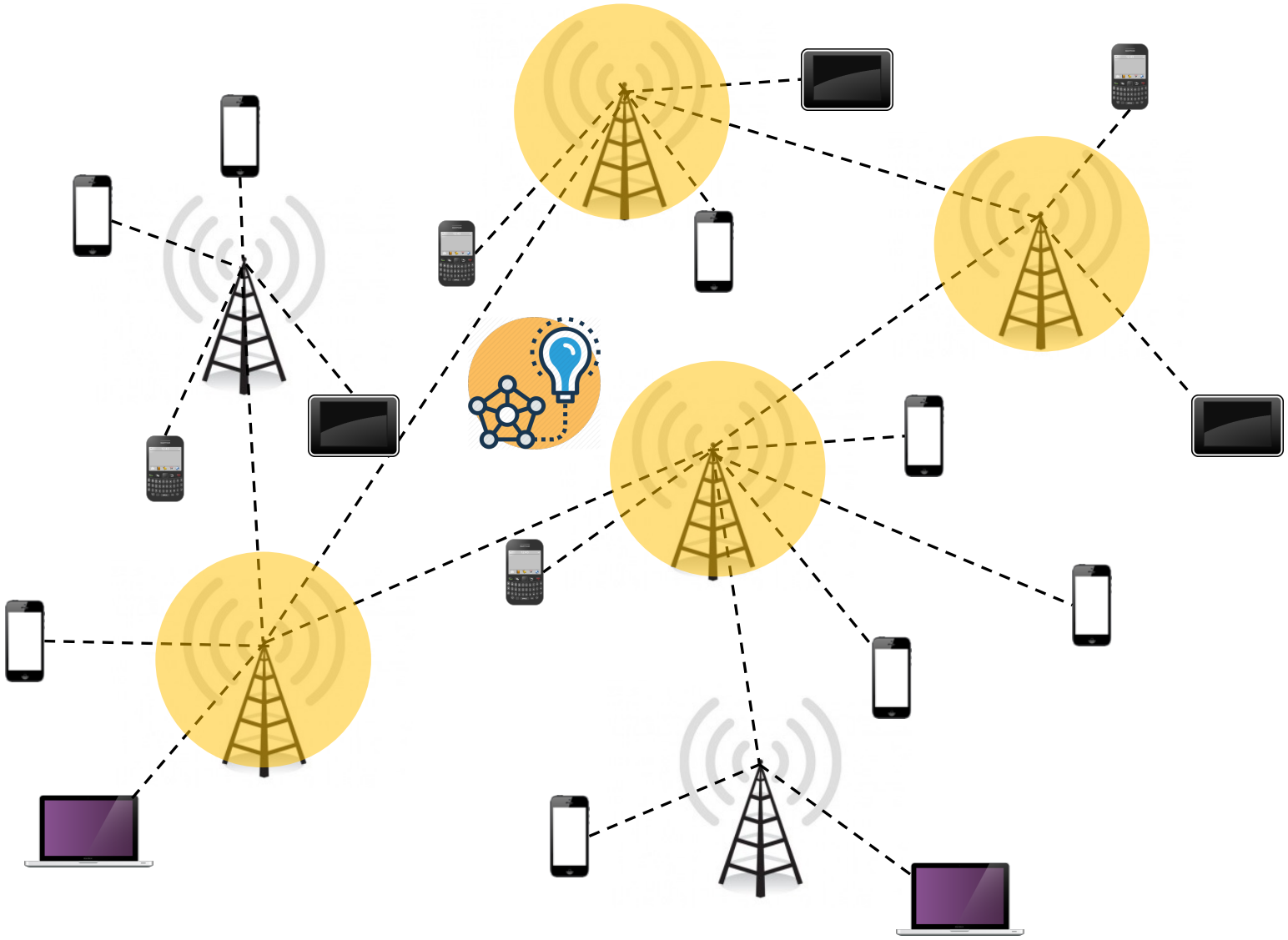


What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How **Carol** does this

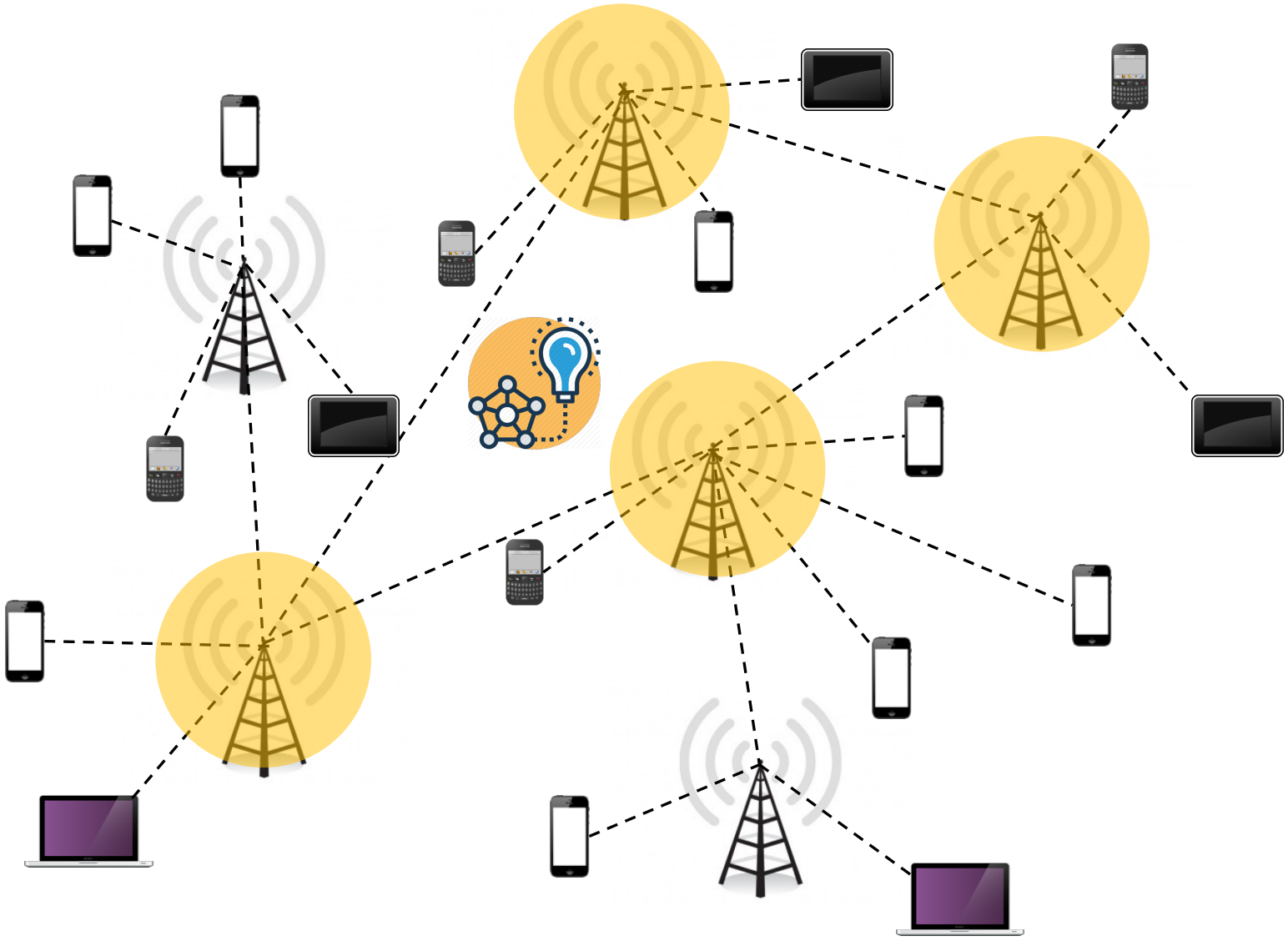


What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How **Carol** does this



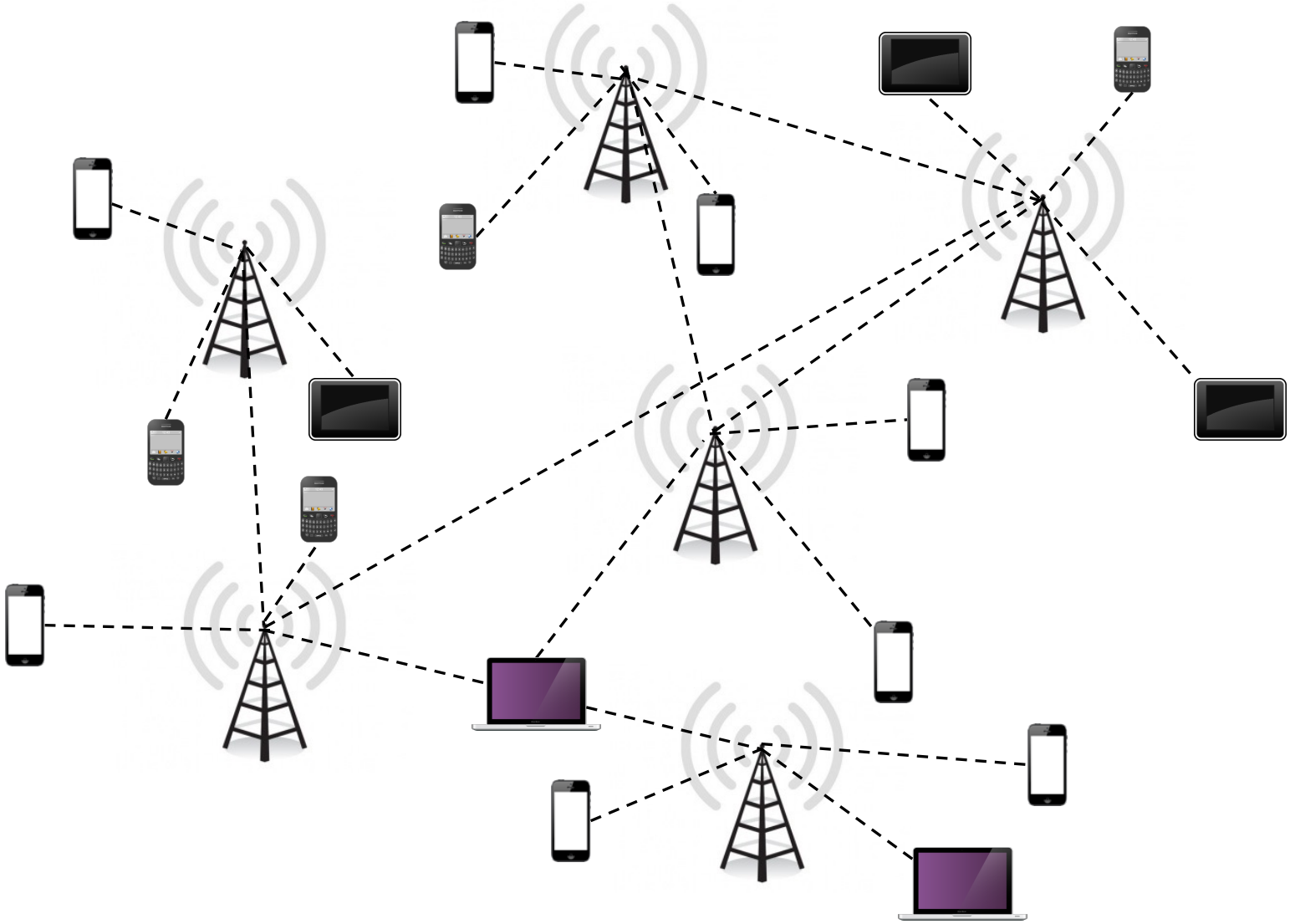
What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this



What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this

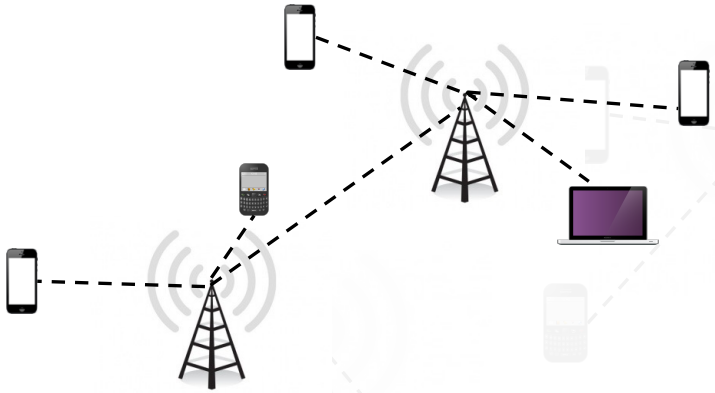
What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this



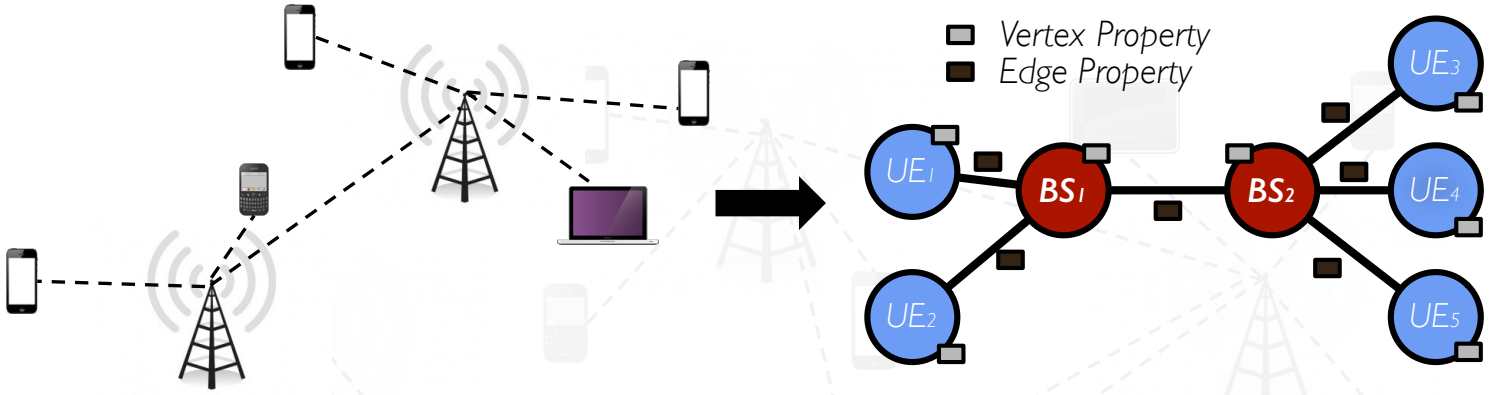
What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this



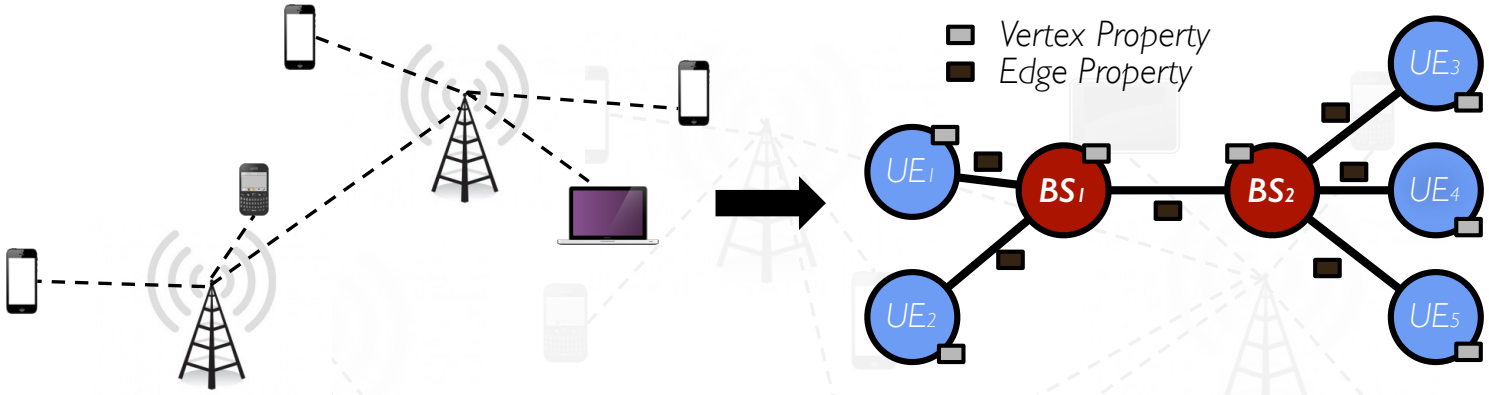
What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this



Interactive ad-hoc analysis on **evolving** graphs

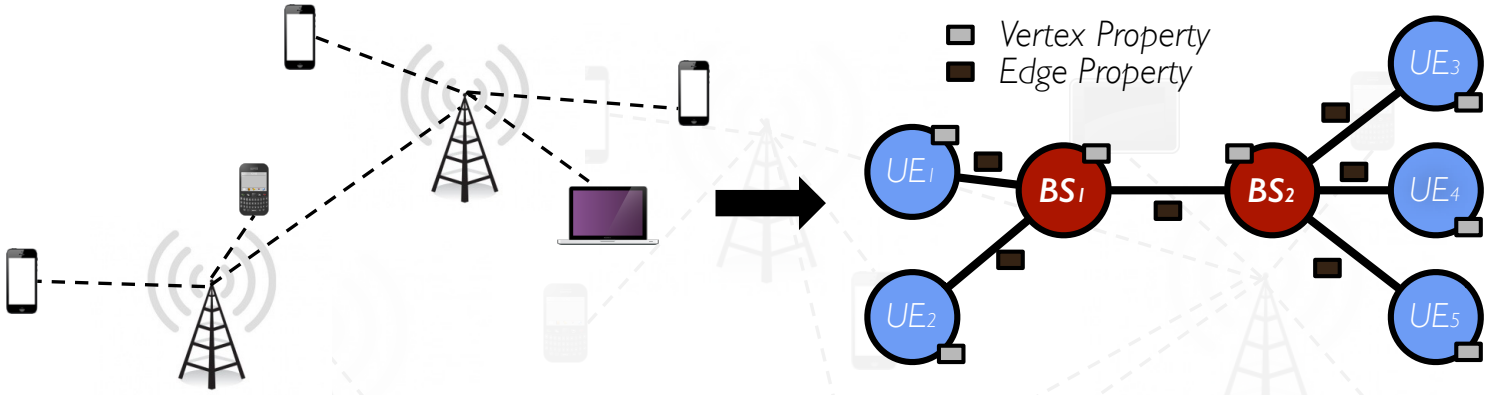
What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this



Interactive ad-hoc analysis on **evolving** graphs

Current systems focus on **streaming** or **temporal** analysis

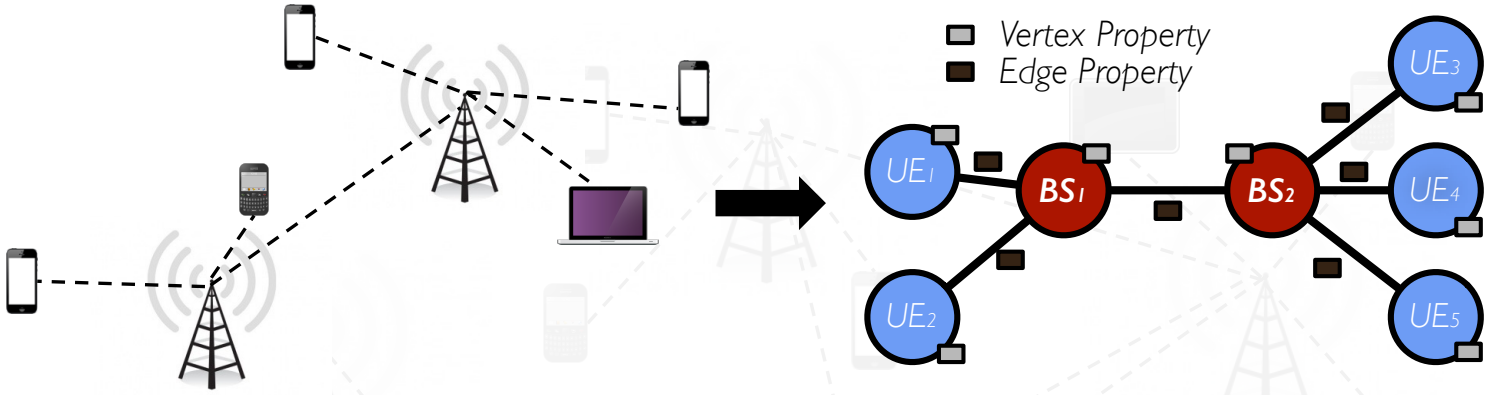
What did the network look like at 9am?

Which towers were congested then?

What was the reason for congestion at these?

How about at 10am?

How **Carol** does this



Interactive ad-hoc analysis on **evolving** graphs

Current systems focus on **streaming** or **temporal** analysis

Can benefit significantly from *efficient* ad-hoc analytics

What did the network look like at 9am?

Which towers were congested then?

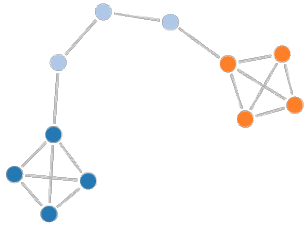
What was the reason for congestion at these?

How about at 10am?

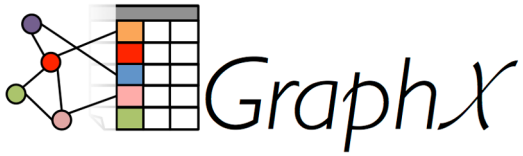
Challenges

Programming
Storage
Performance

How do we enable ad-hoc queries on dynamic graphs in a **natural** and **intuitive** way?



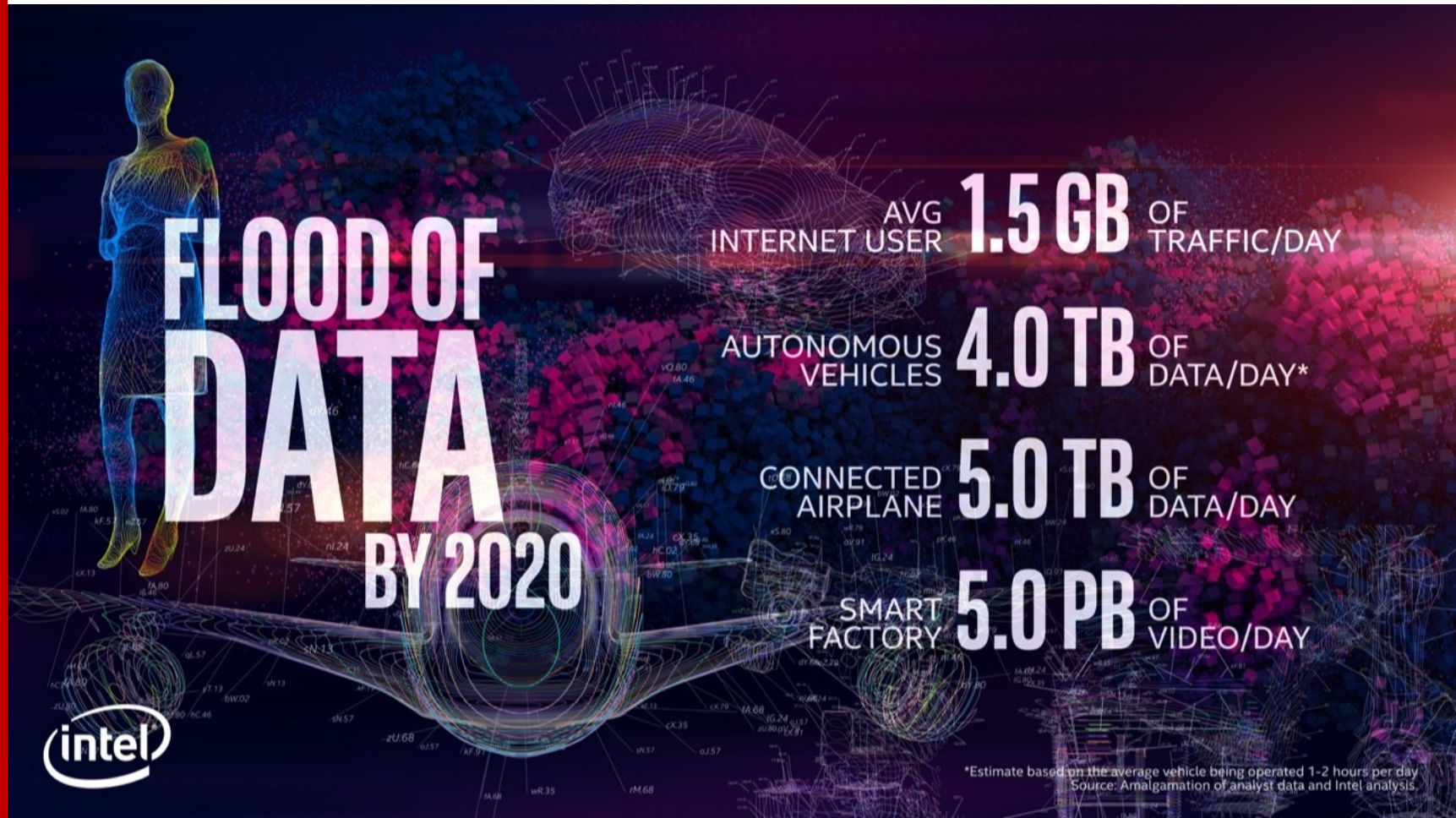
NetworkX



Challenges

Programming
Storage
Performance

Ad-hoc queries are
data intensive



Challenges

Programming
Storage
Performance

Ad-hoc queries are both
interactive and **exploratory**

Changes to the graph are **relatively small *during*** ad-hoc analysis

Changes to the graph are **relatively small *during*** ad-hoc analysis

Queries are frequently applied to **multiple windows *close-by*** in time

Changes to the graph are **relatively small *during*** ad-hoc analysis

Queries are frequently applied to **multiple windows *close-by*** in time

TEGRA accelerates ad-hoc analytics by **sharing storage and computation** across queries and windows

Challenges

Programming

Timelapse abstraction with simple API

Storage

Incremental storage for efficient access

Computation

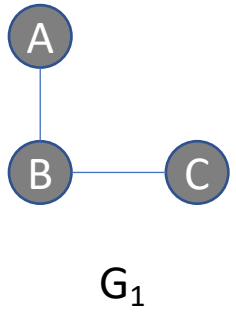
Incremental computation model for interactivity & efficiency

Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots

Timelapse Abstraction

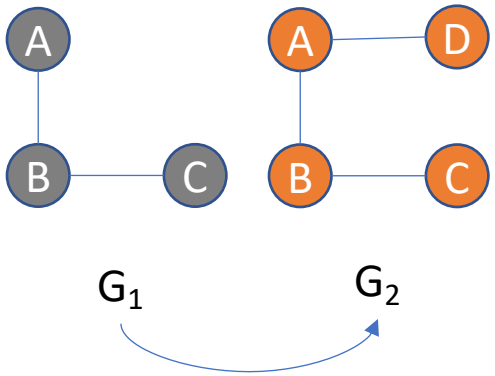
Key idea: Illusion of a **series** of static graph snapshots



G_1

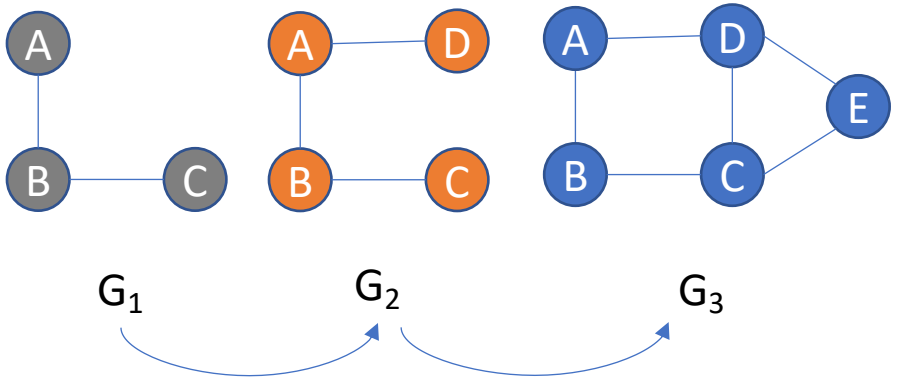
Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots



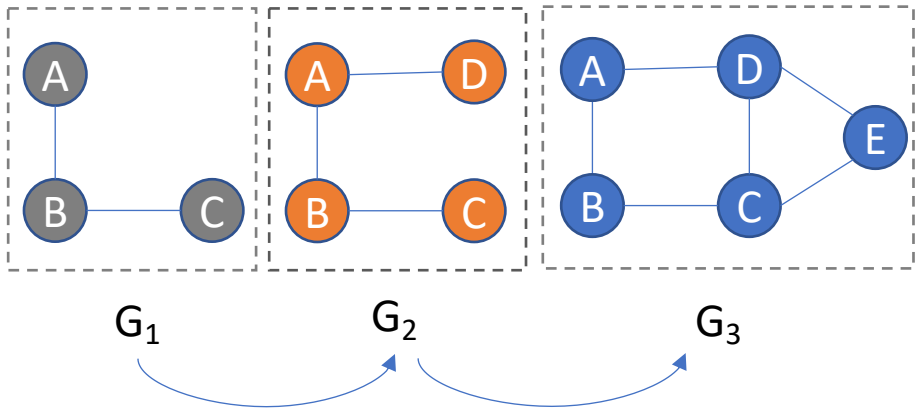
Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots



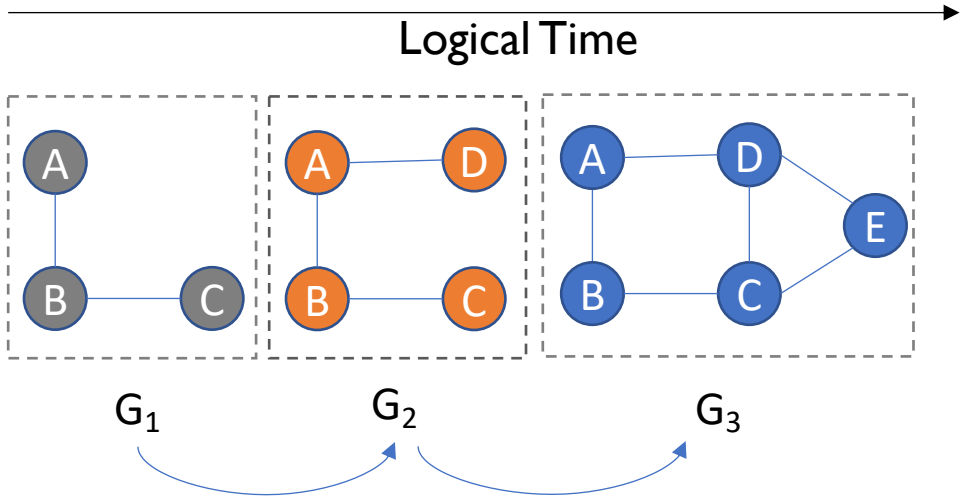
Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots



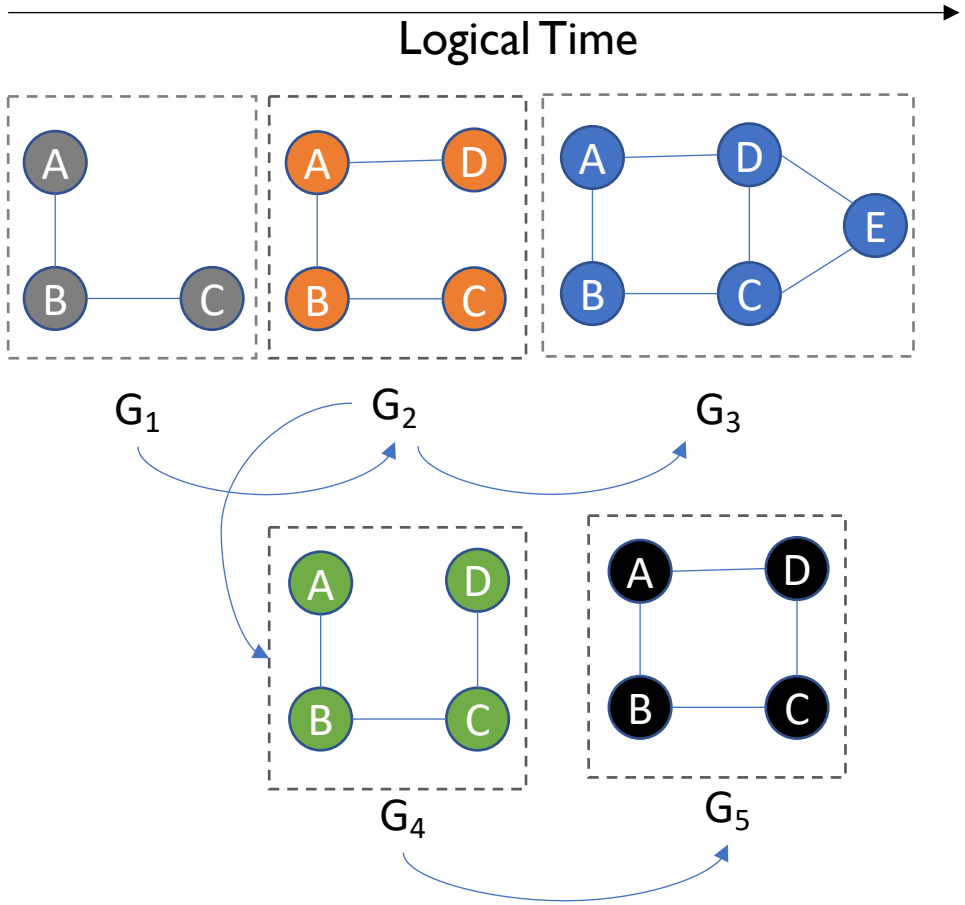
Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots



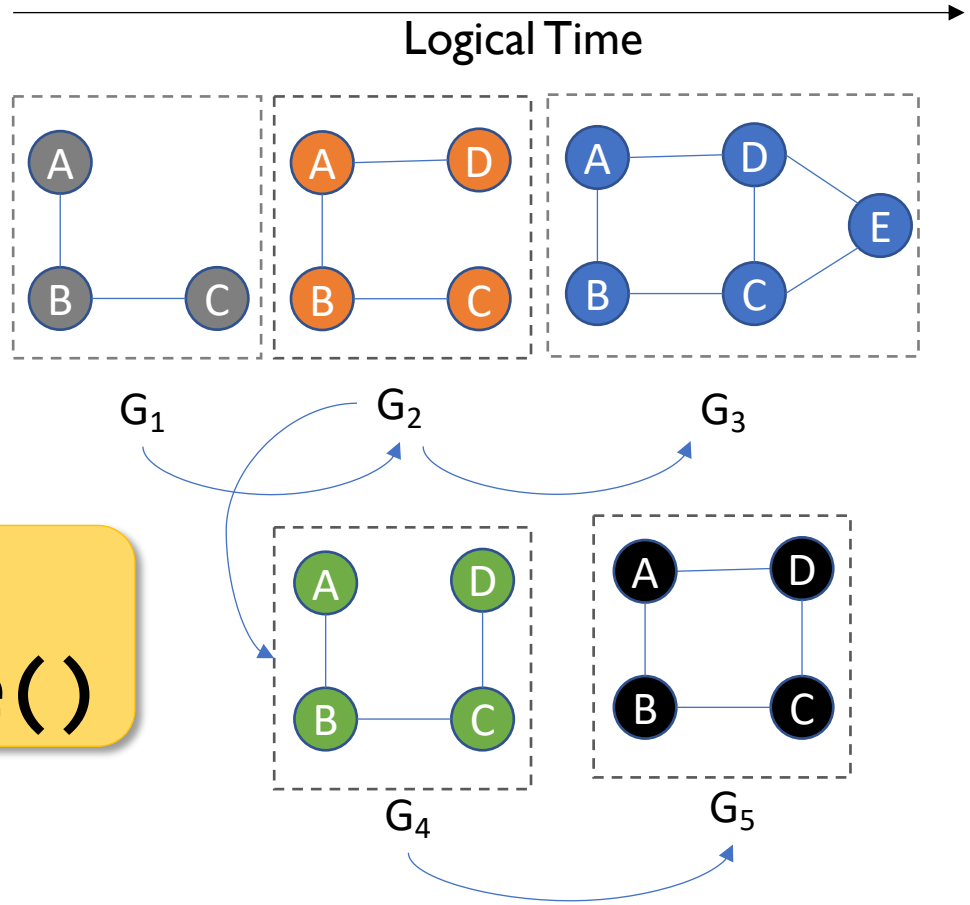
Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots



Timelapse Abstraction

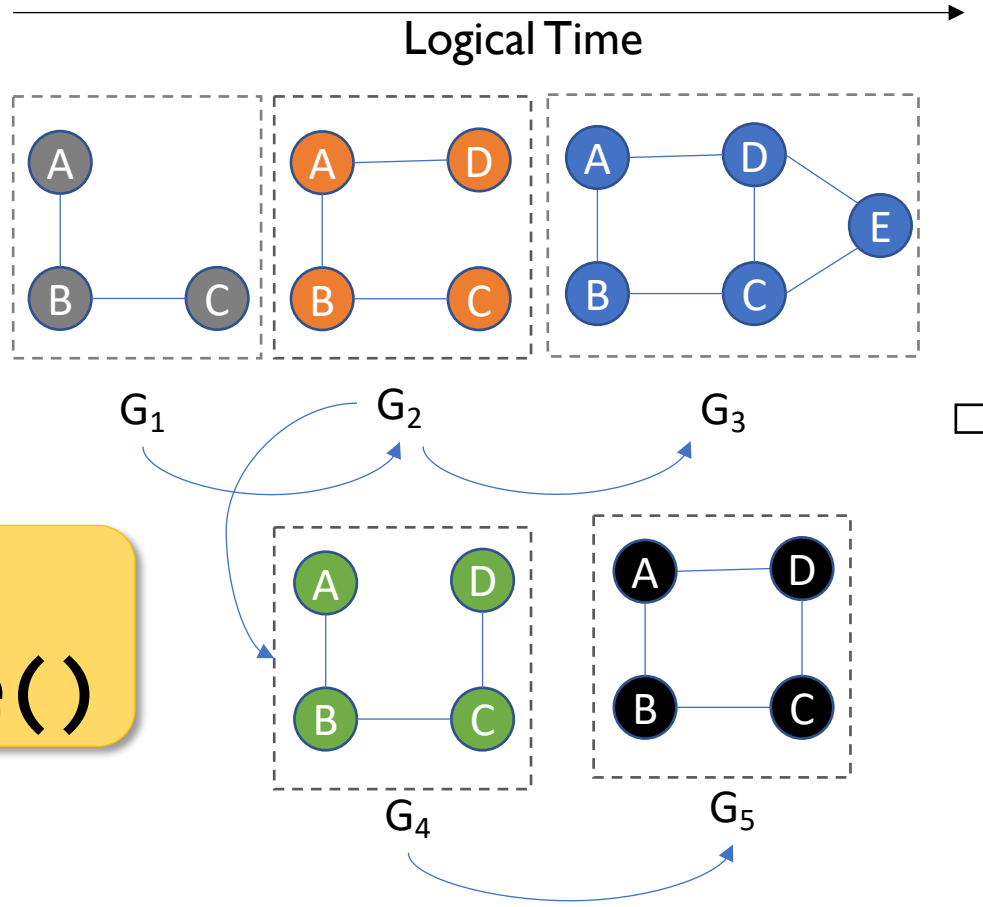
Key idea: Illusion of a **series** of static graph snapshots

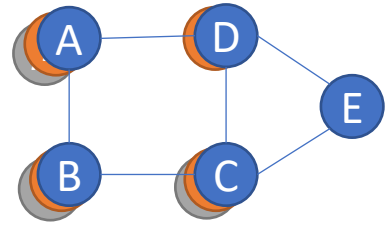



```
save()
retrieve()
```

Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots



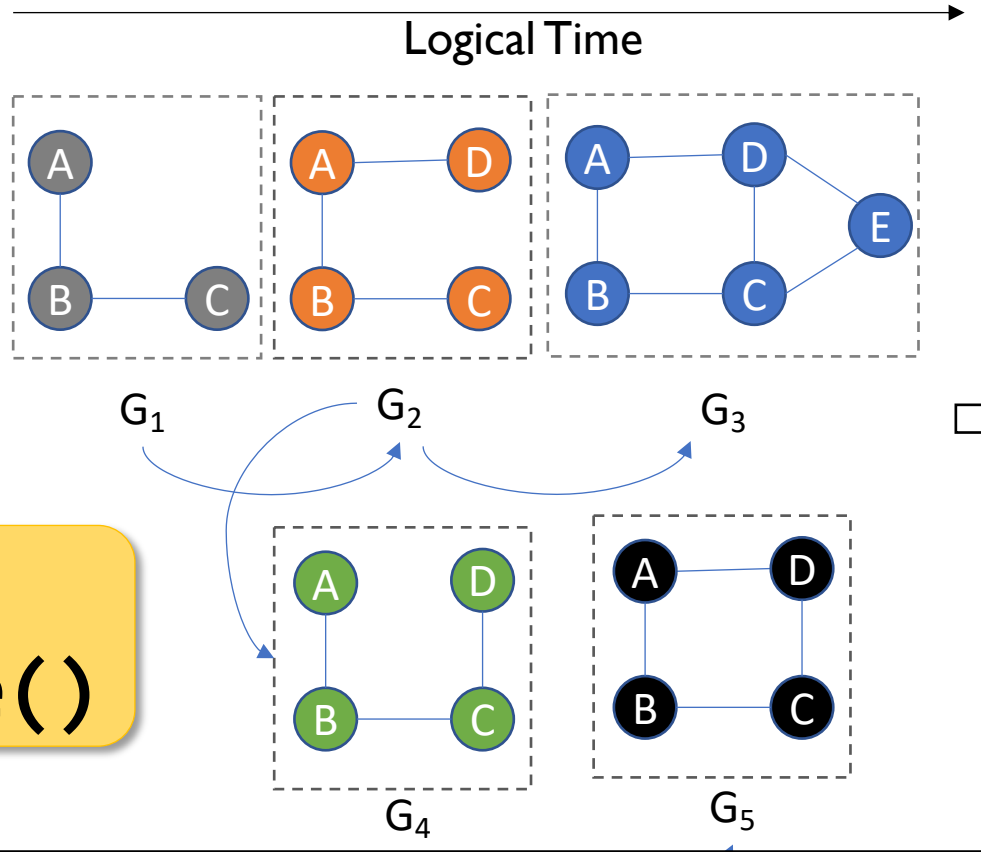


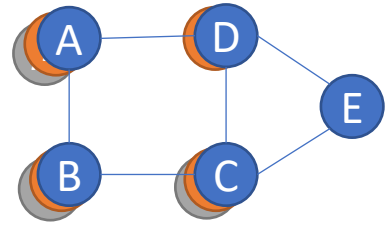
`save()`
`retrieve()`

`diff()`
`merge()`
`expand()`

Timelapse Abstraction

Key idea: Illusion of a **series** of static graph snapshots





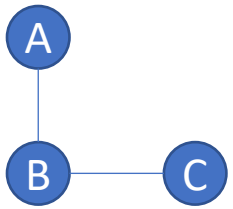
`save()`
`retrieve()`

`diff()`
`merge()`
`expand()`

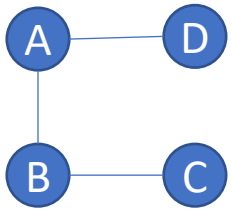
Enable system to **efficiently store & operate** on them

Snapshot **Storage**

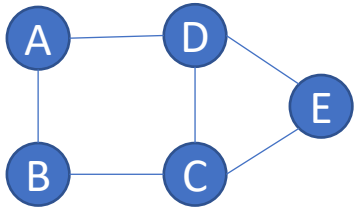
Store entire snapshots



G_1



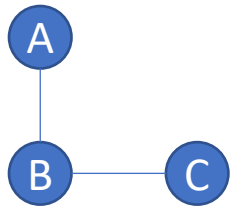
G_2



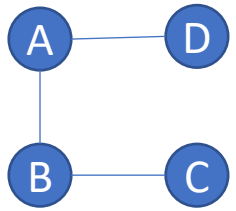
G_3

Snapshot **Storage**

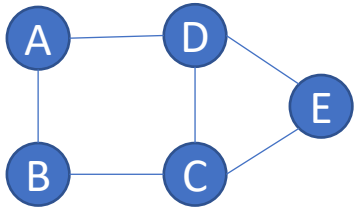
Store entire snapshots



G_1



G_2

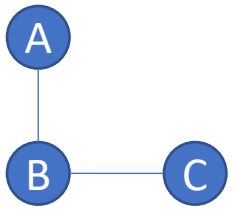


G_3

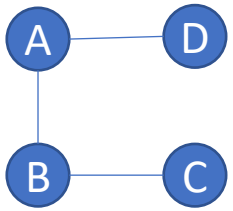
+ Efficient retrieval
- Storage overhead

Snapshot **Storage**

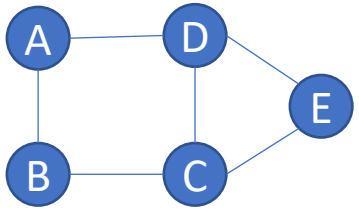
Store entire snapshots



G_1



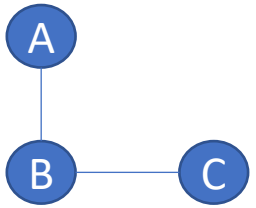
G_2



G_3

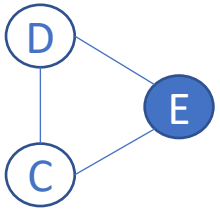
+ Efficient retrieval
- Storage overhead

Store only differences



δg_1

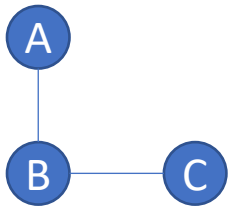
δg_2



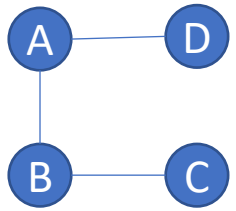
δg_3

Snapshot **Storage**

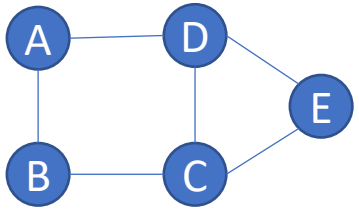
Store entire snapshots



G_1



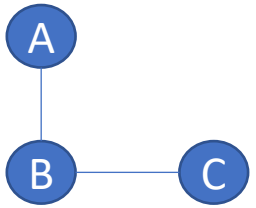
G_2



G_3

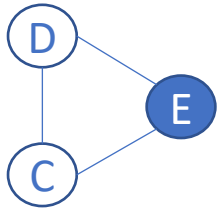
+ Efficient retrieval
- Storage overhead

Store only differences



δg_1

δg_2

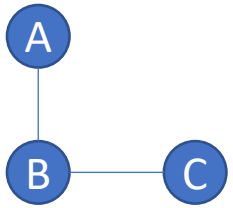


δg_3

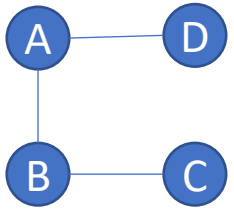
+ Efficient storage
- Retrieval overhead

Snapshot **Storage**

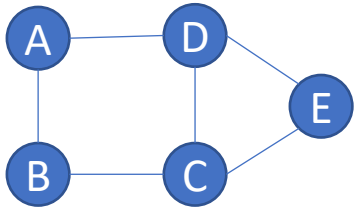
Store entire snapshots



G_1



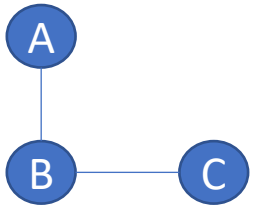
G_2



G_3

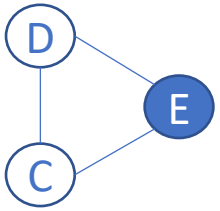
+ Efficient retrieval
- Storage overhead

Store only differences



δg_1

δg_2



δg_3

+ Efficient storage
- Retrieval overhead

Fundamental trade-off between storage **overhead** and retrieval **efficiency**

Challenges

Programming

Timelapse abstraction with simple API

Storage

Incremental storage for efficient access

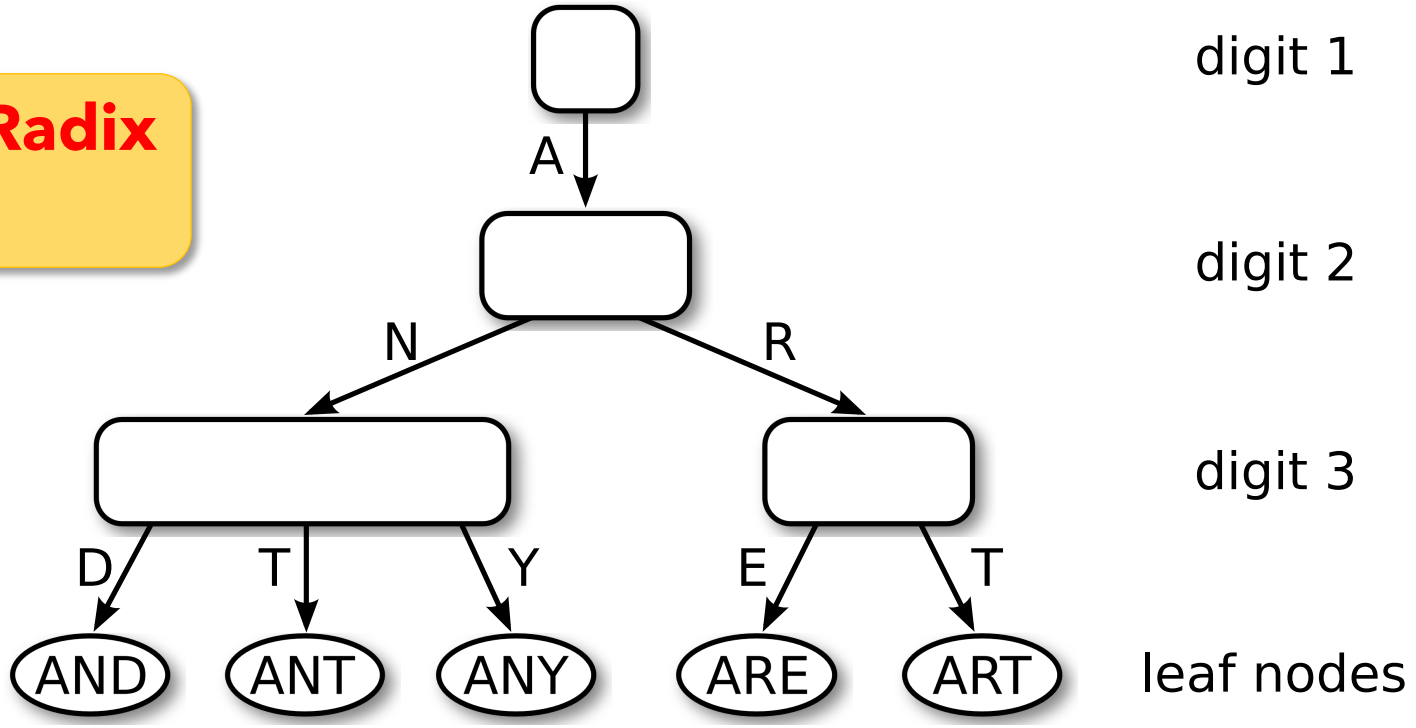
Computation

Incremental computation model for interactivity & efficiency

Optimizing **Storage**

Key idea: Leverage persistent data structures

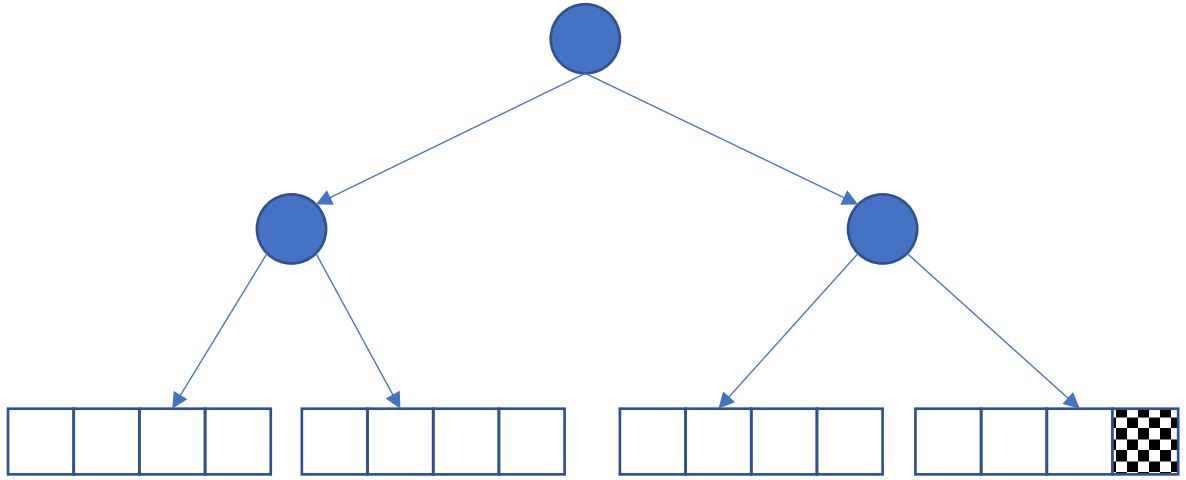
Adaptive Radix Tree



"The adaptive radix tree: ARTful indexing for main-memory databases", Leis et. al., ICDE 2013

Optimizing **Storage**

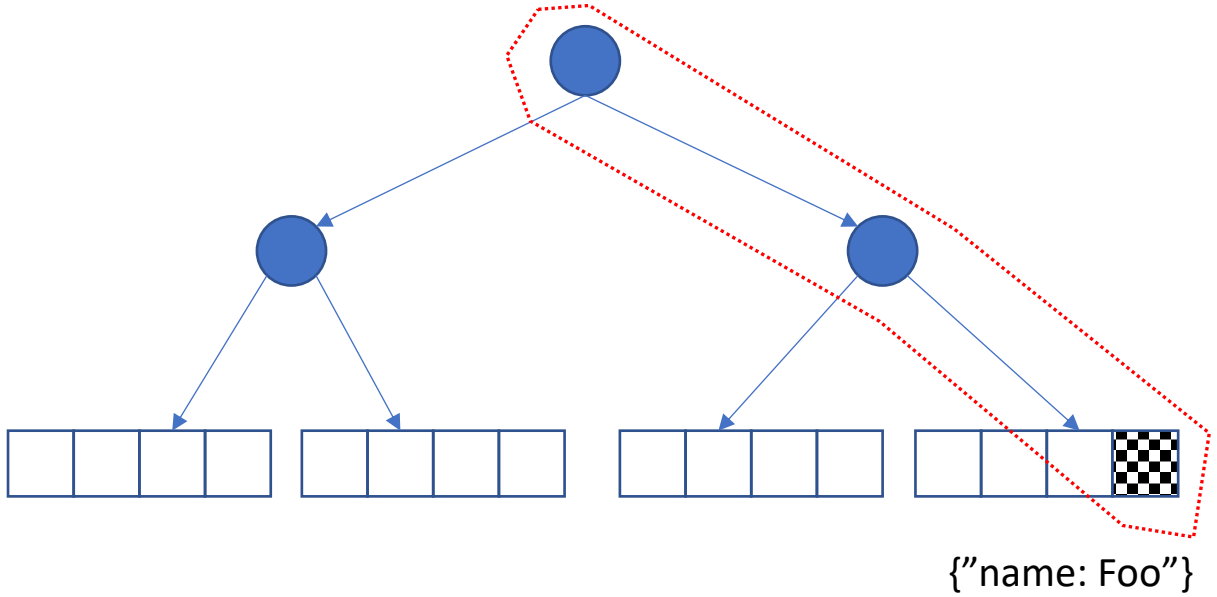
Key idea: Build a persistent data structure based distributed dynamic graph store



Vertex ➡ Property

Optimizing **Storage**

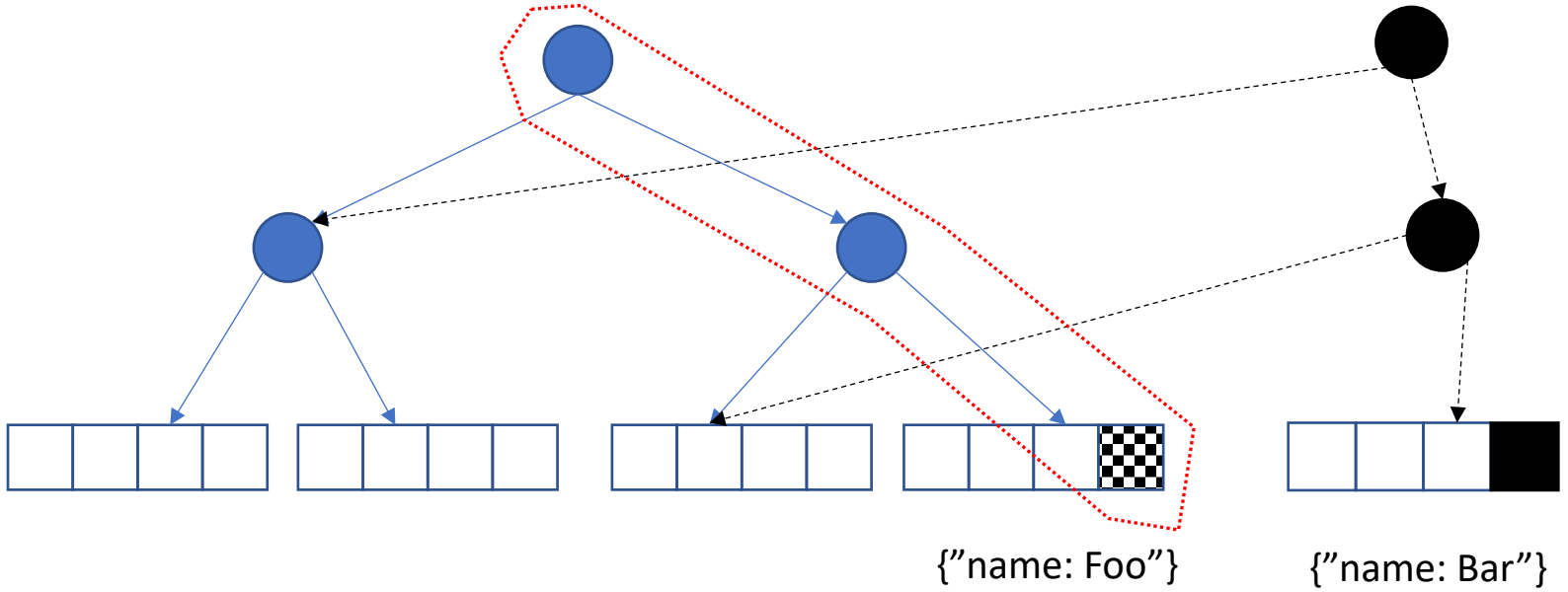
Key idea: Build a persistent data structure based distributed dynamic graph store



Vertex➡Property

Optimizing **Storage**

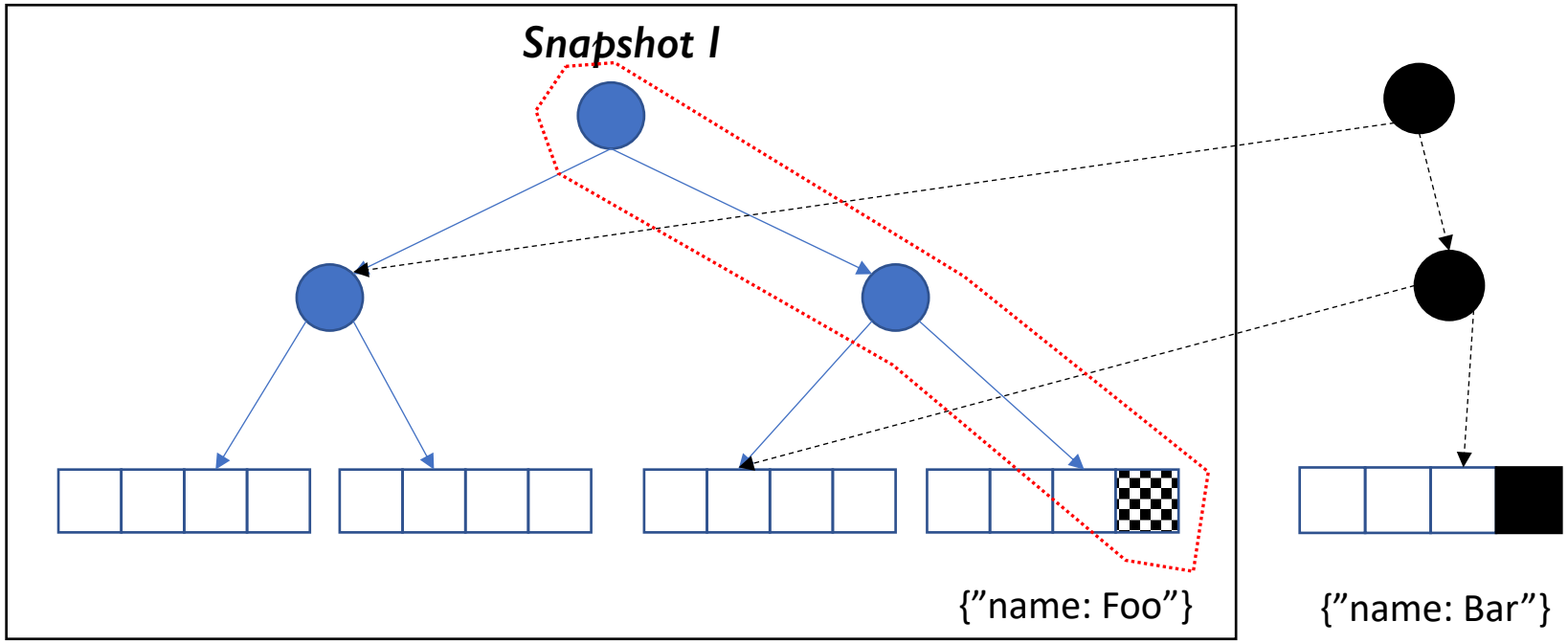
Key idea: Build a persistent data structure based distributed dynamic graph store



Vertex ➡ Property

Optimizing **Storage**

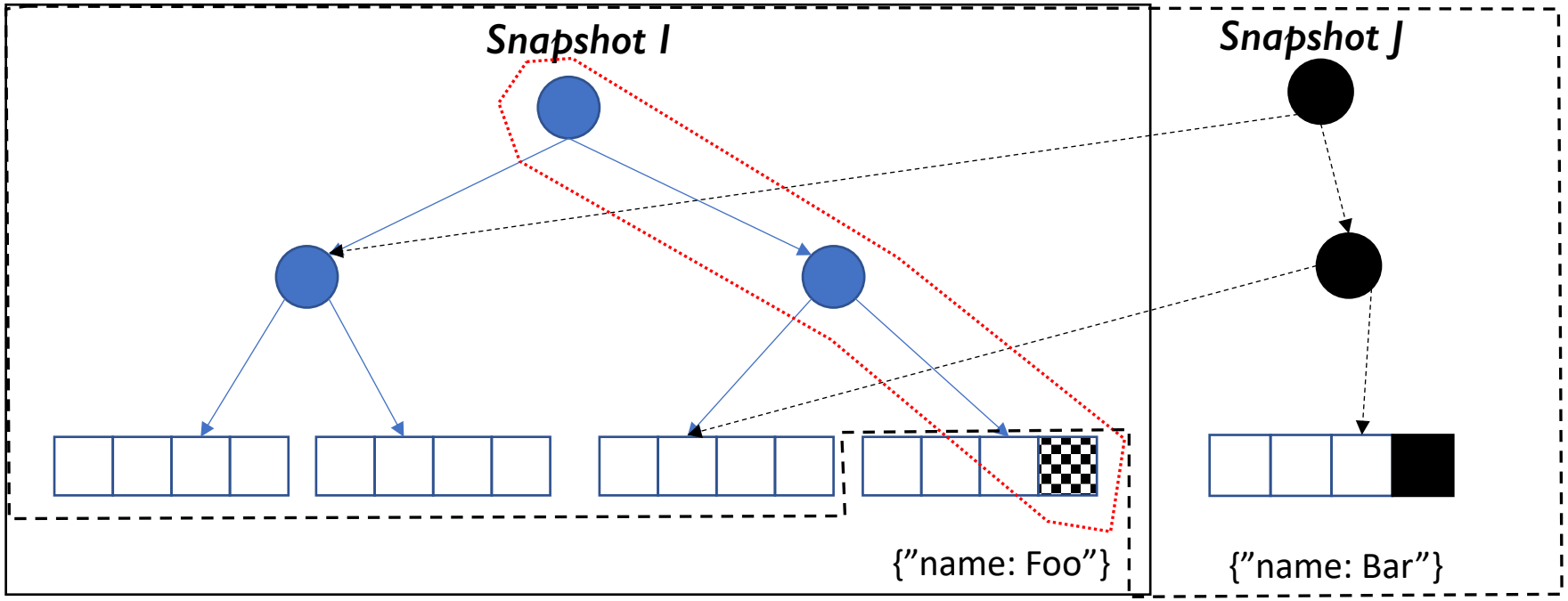
Key idea: Build a persistent data structure based distributed dynamic graph store



Vertex ➡ Property

Optimizing **Storage**

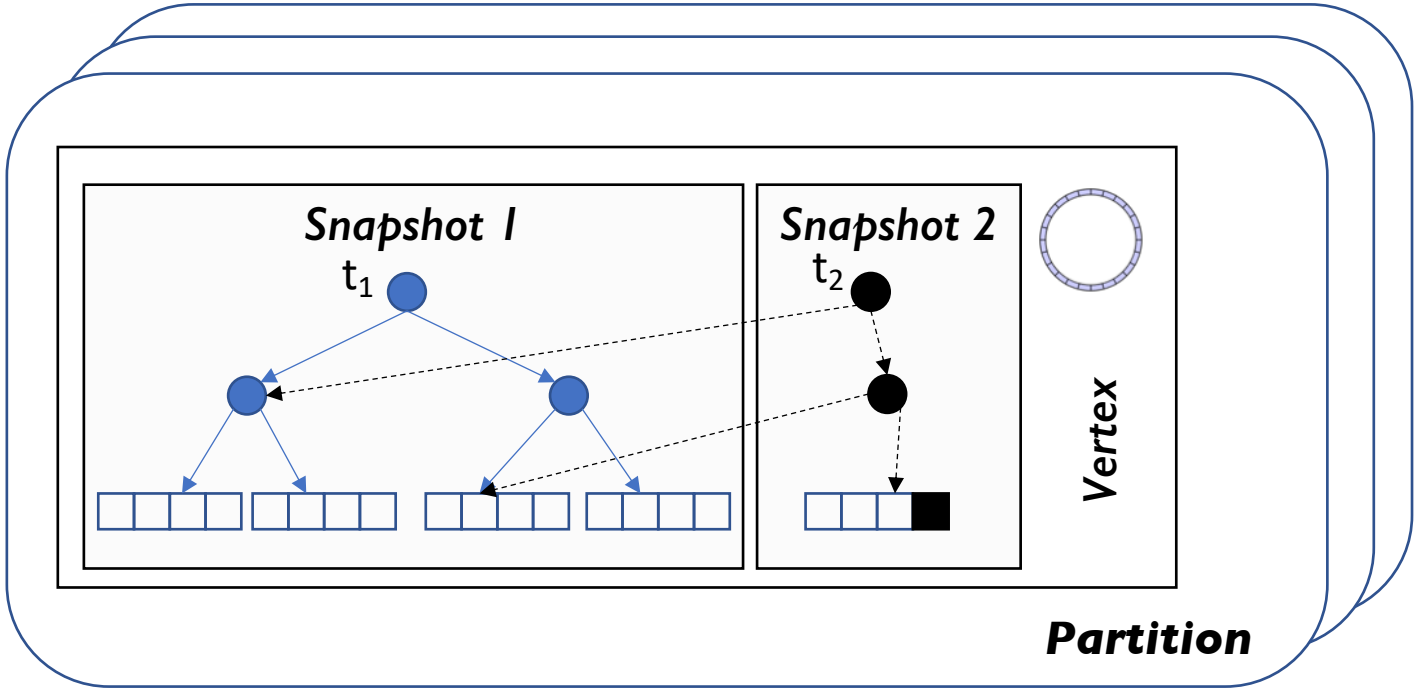
Key idea: Build a persistent data structure based distributed dynamic graph store



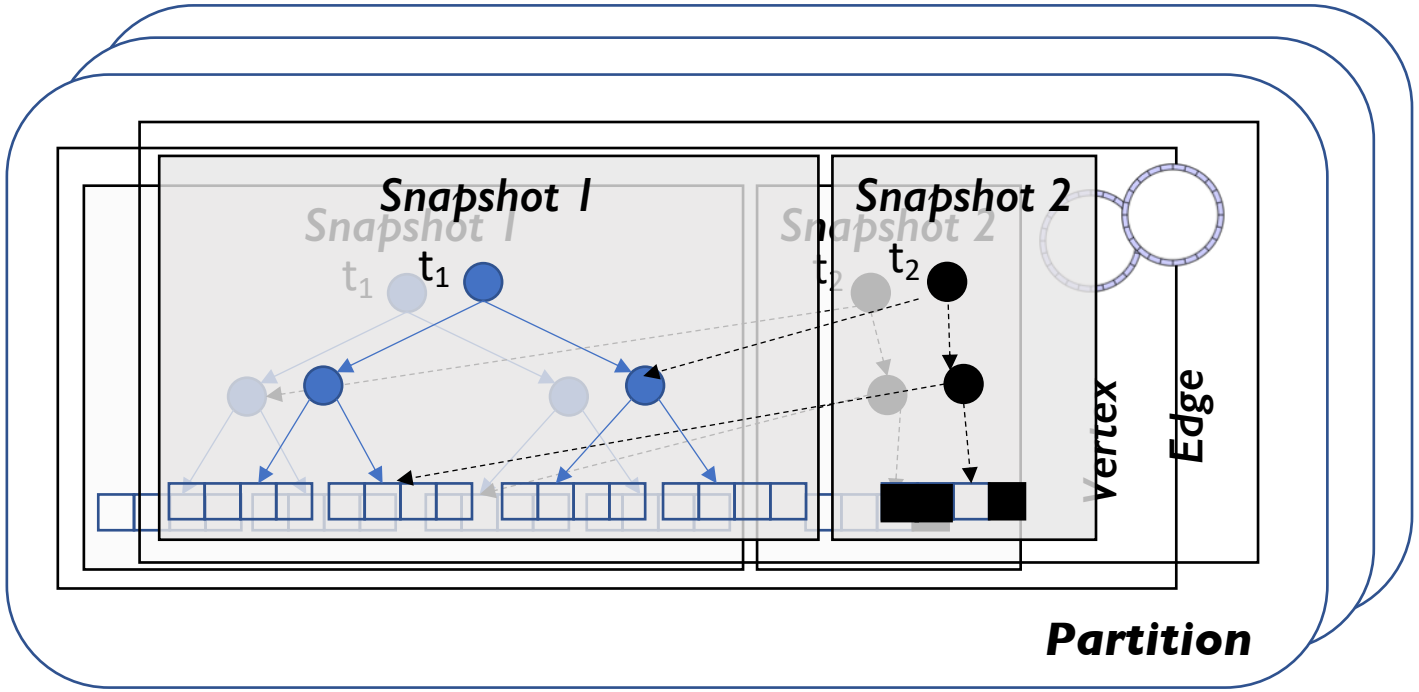
Vertex ➡ Property

Distributed Graph Snapshot Index (DGSI)

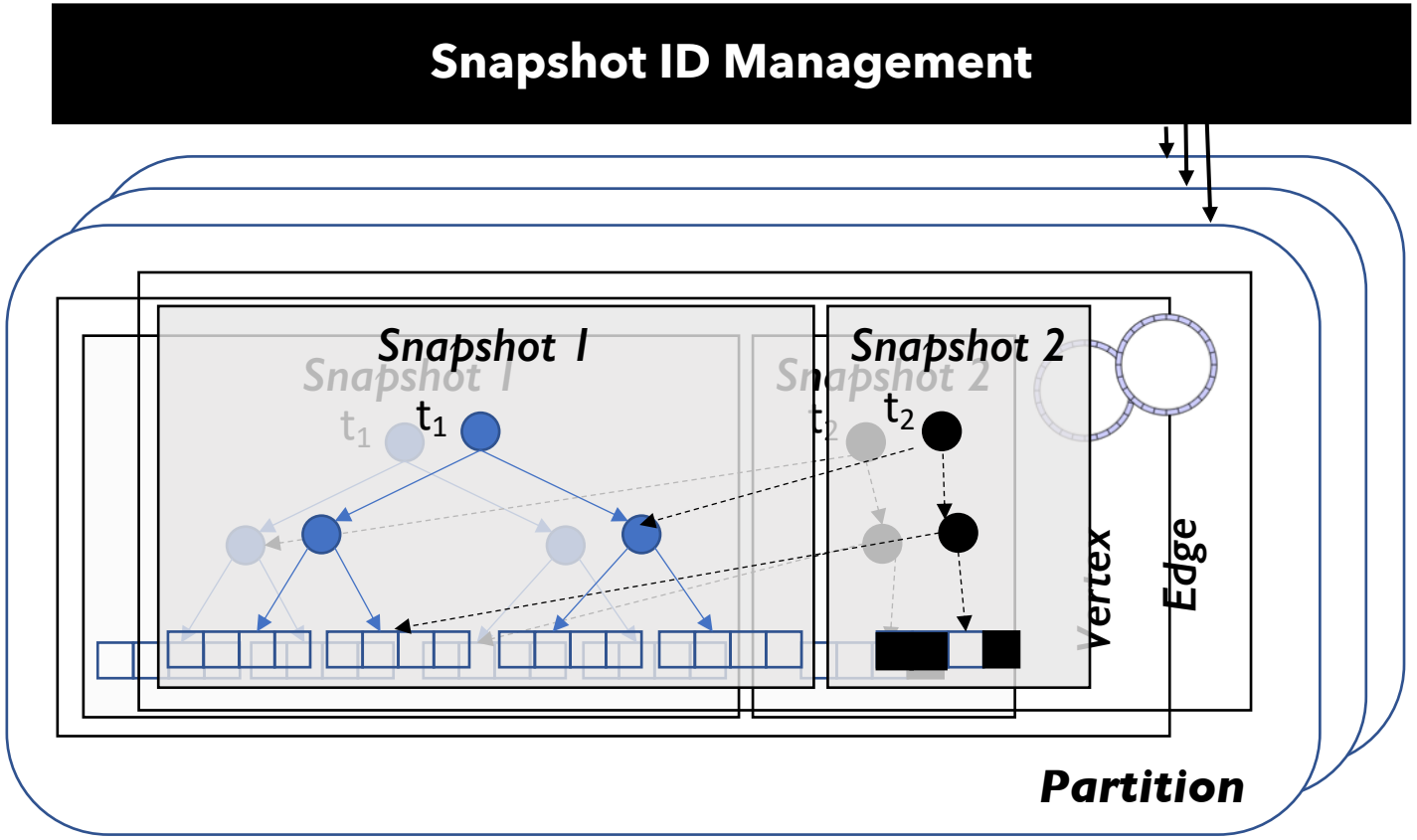
Distributed Graph Snapshot Index (DGSI)



Distributed Graph Snapshot Index (DGSI)



Distributed Graph Snapshot Index (DGSI)

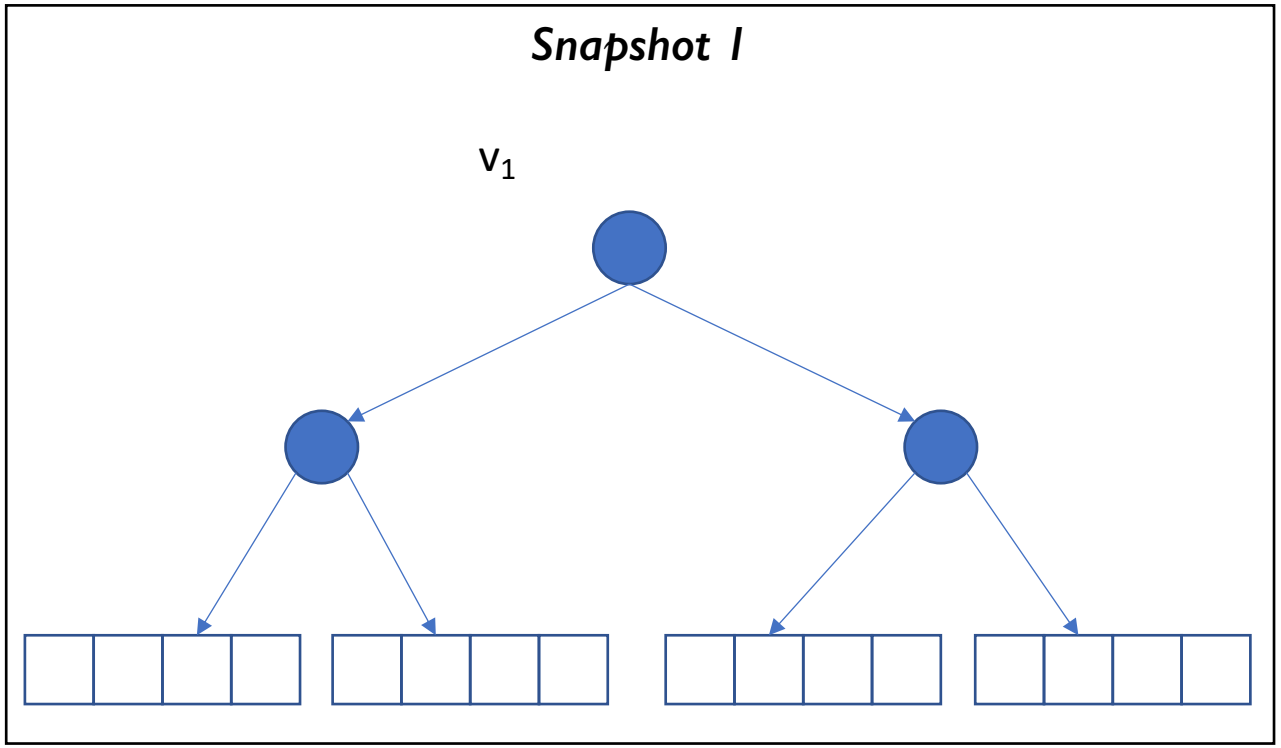


Memory Management

Enables management of storage at the **leaf** level

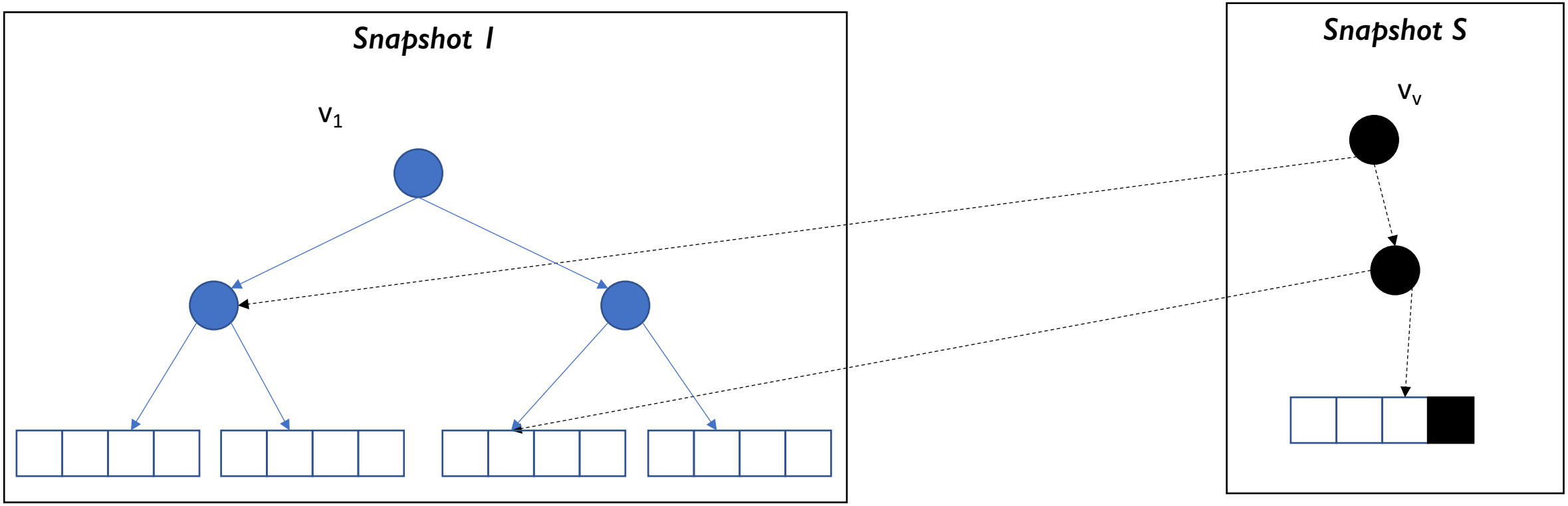
Memory Management

Enables management of storage at the **leaf** level



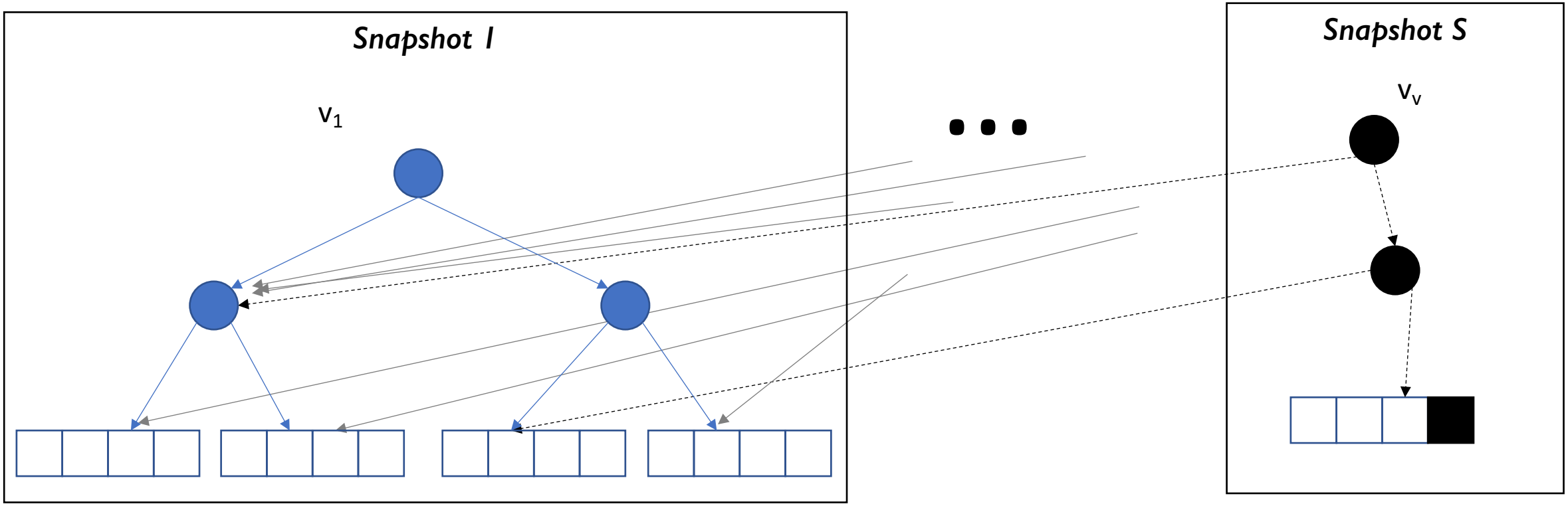
Memory Management

Enables management of storage at the **leaf** level



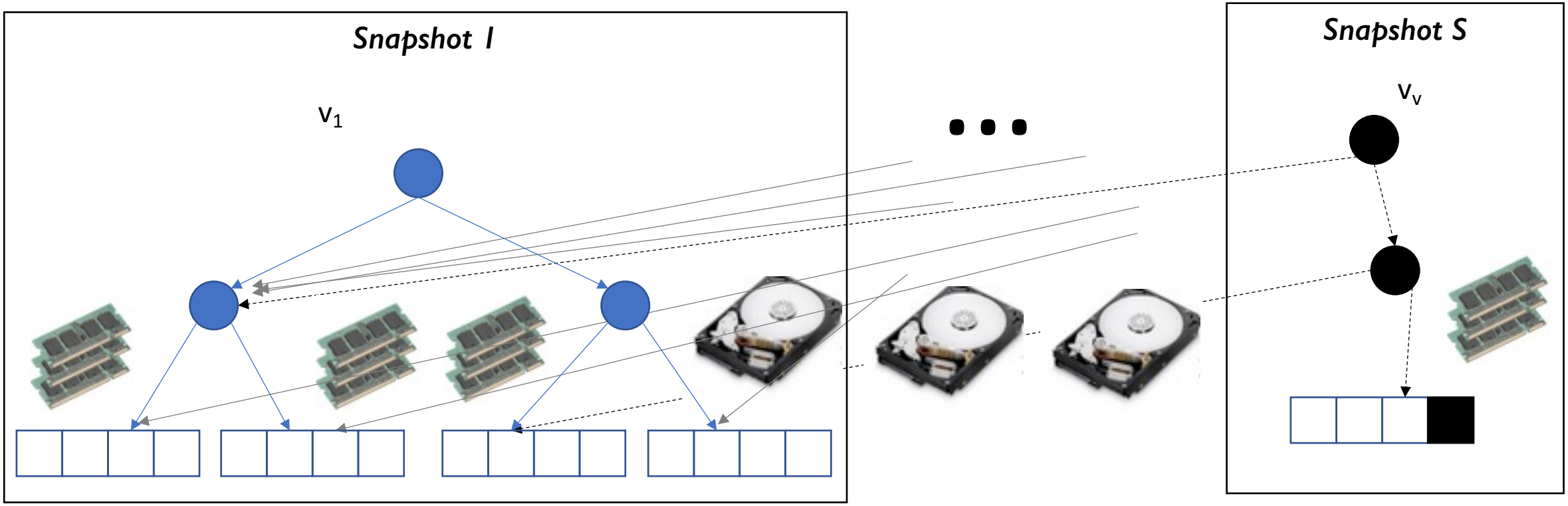
Memory Management

Enables management of storage at the **leaf** level



Memory Management

Enables management of storage at the **leaf** level



Challenges

Programming

Timelapse abstraction with simple API

Storage

Incremental storage for efficient access

Computation

Incremental computation model for interactivity & efficiency

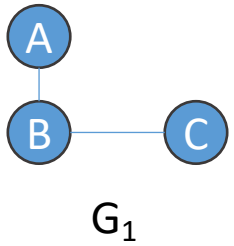
Computation Model

Leverages the characteristics of graph-parallel execution model

Computation Model

Leverages the characteristics of graph-parallel execution model

Graph

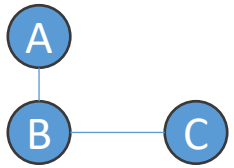


G_1

Computation Model

Leverages the characteristics of graph-parallel execution model

Graph



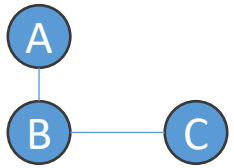
G_1

Computation

Computation Model

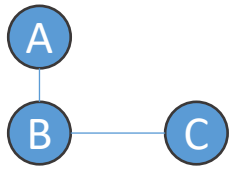
Leverages the characteristics of graph-parallel execution model

Graph



G_1

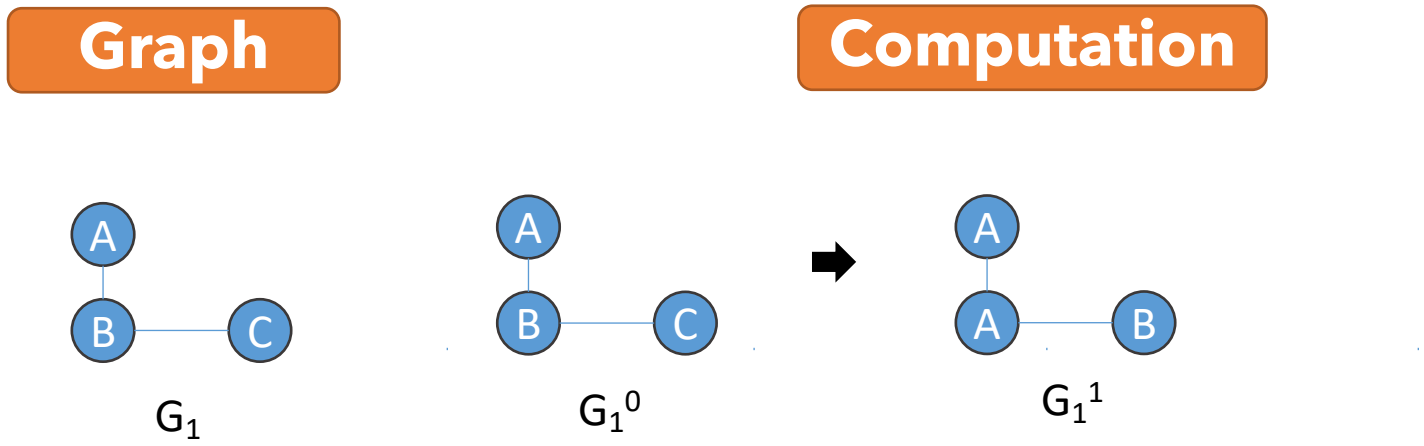
Computation



G_1^0

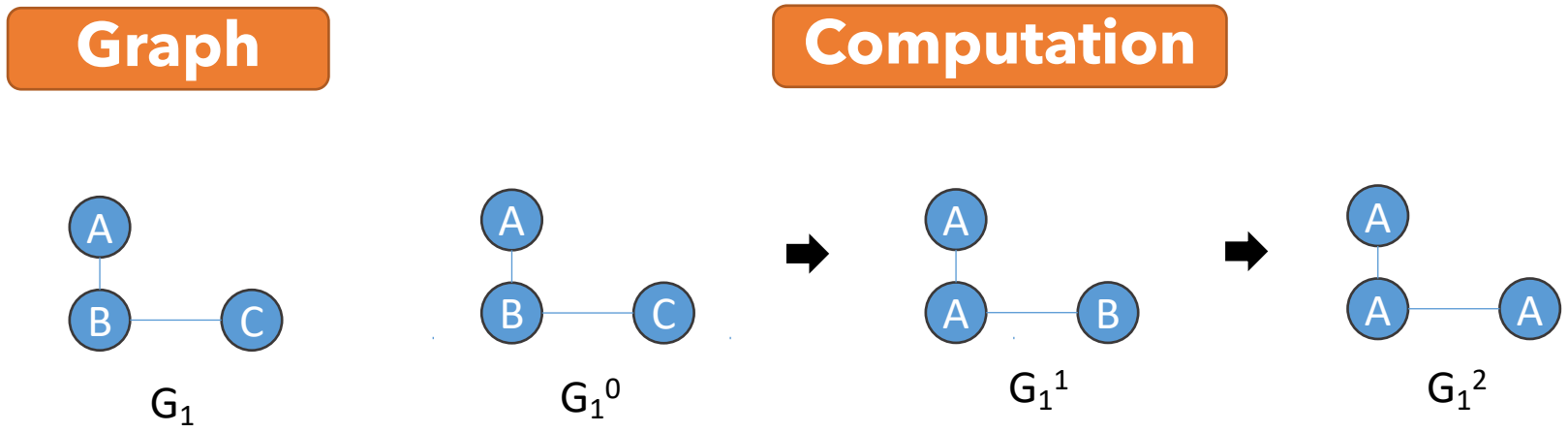
Computation Model

Leverages the characteristics of graph-parallel execution model



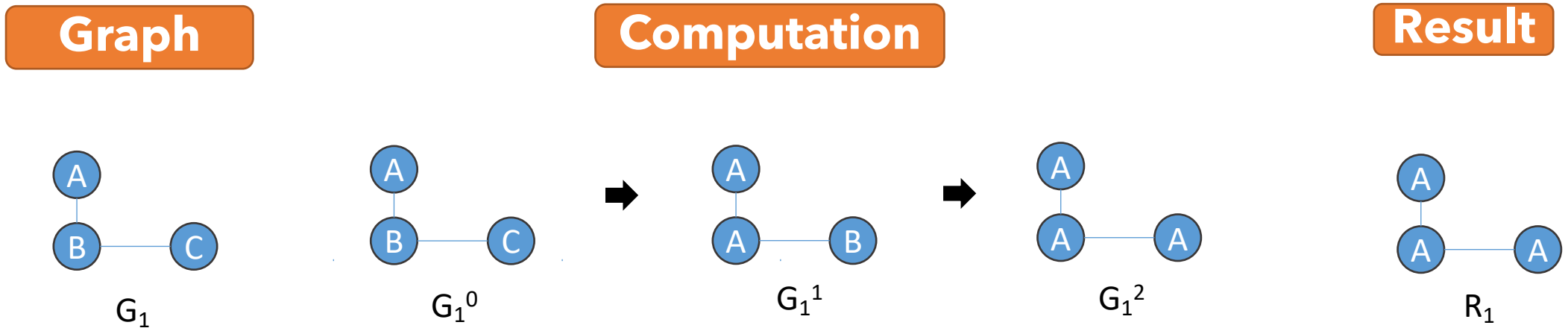
Computation Model

Leverages the characteristics of graph-parallel execution model



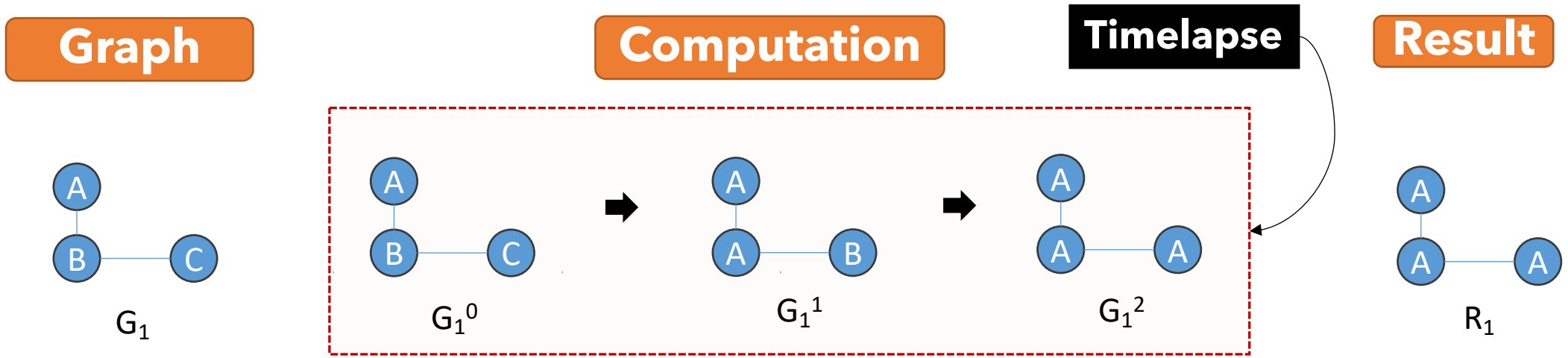
Computation Model

Leverages the characteristics of graph-parallel execution model



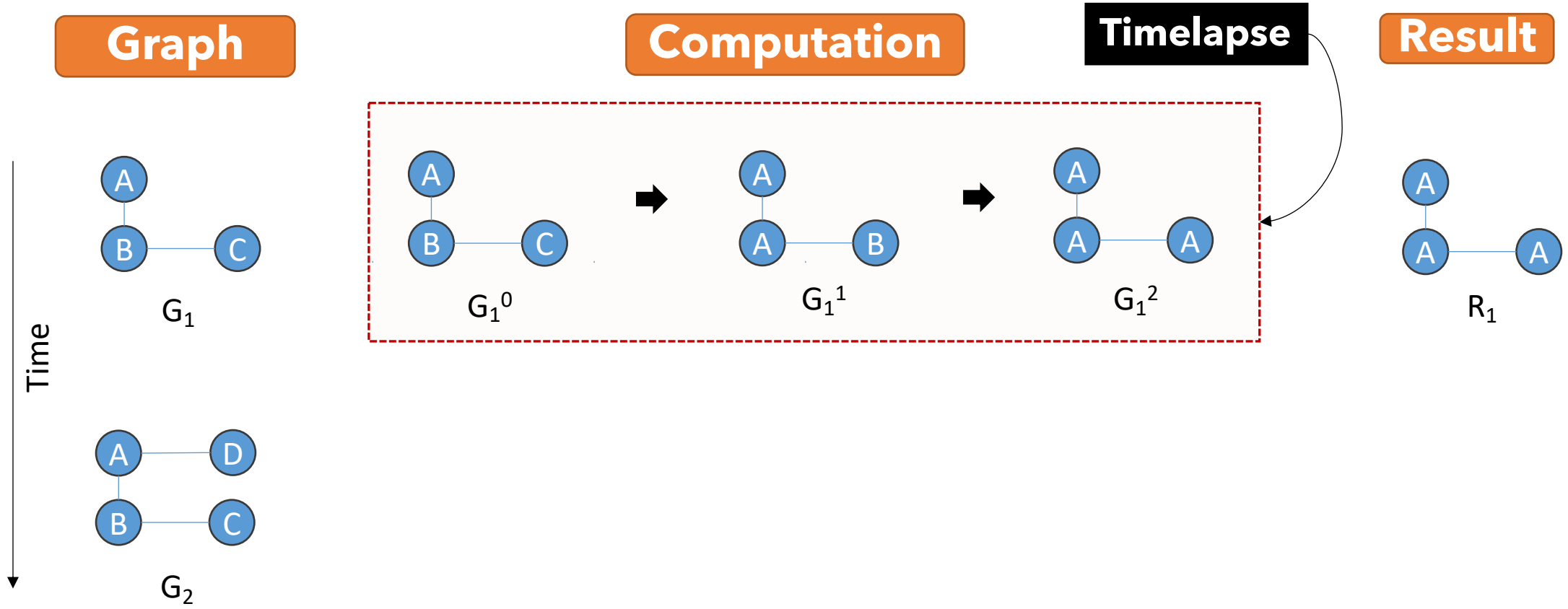
Computation Model

Leverages the characteristics of graph-parallel execution model



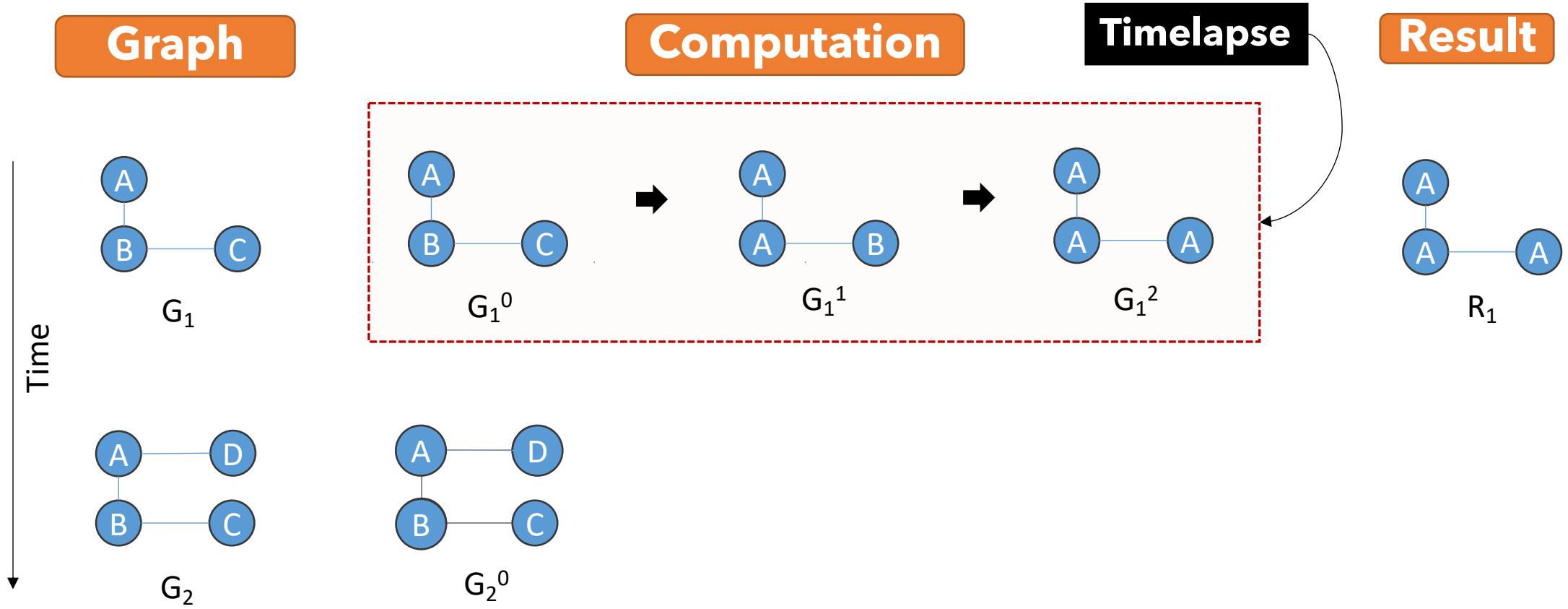
Computation Model

Leverages the characteristics of graph-parallel execution model



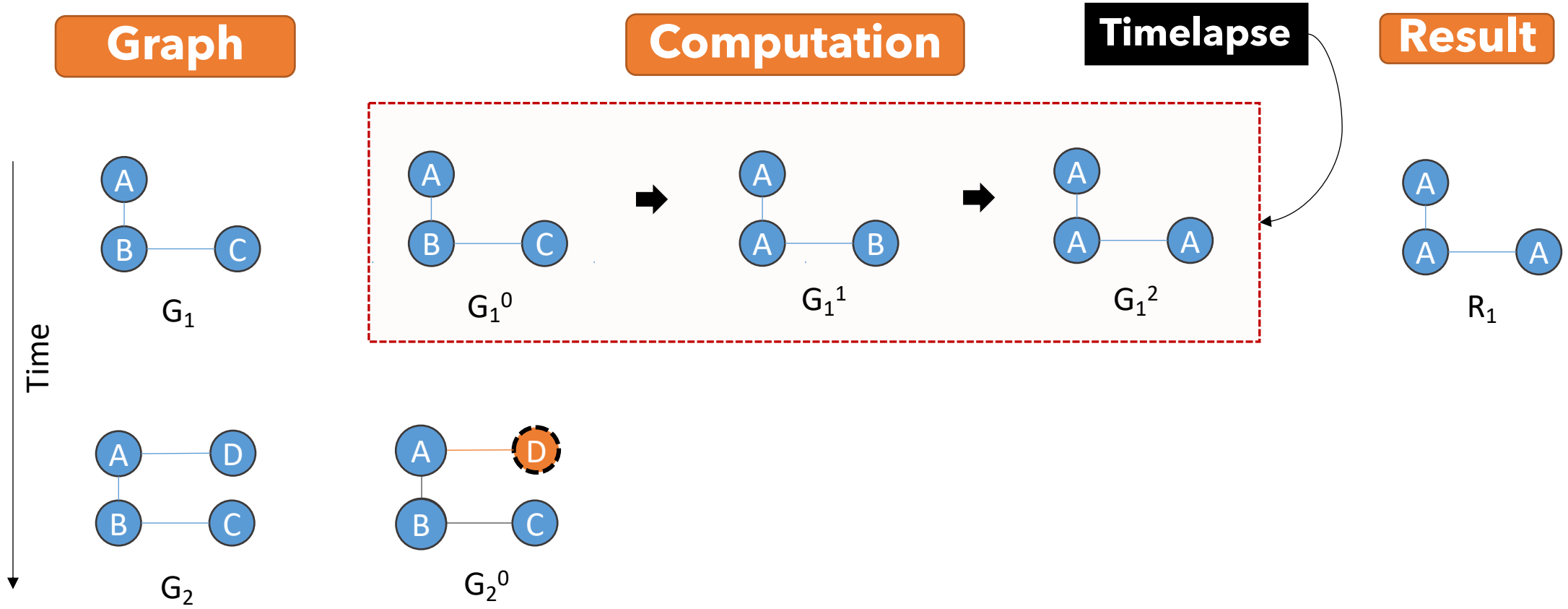
Computation Model

Leverages the characteristics of graph-parallel execution model



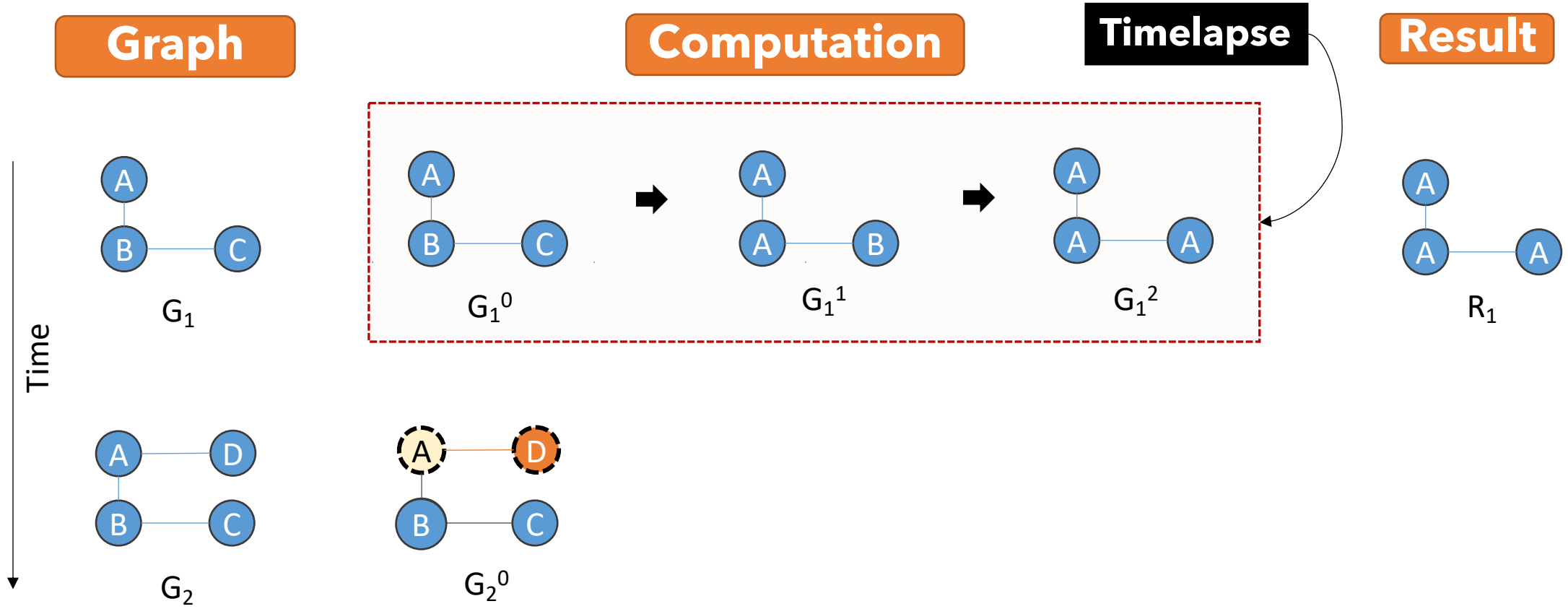
Computation Model

Leverages the characteristics of graph-parallel execution model



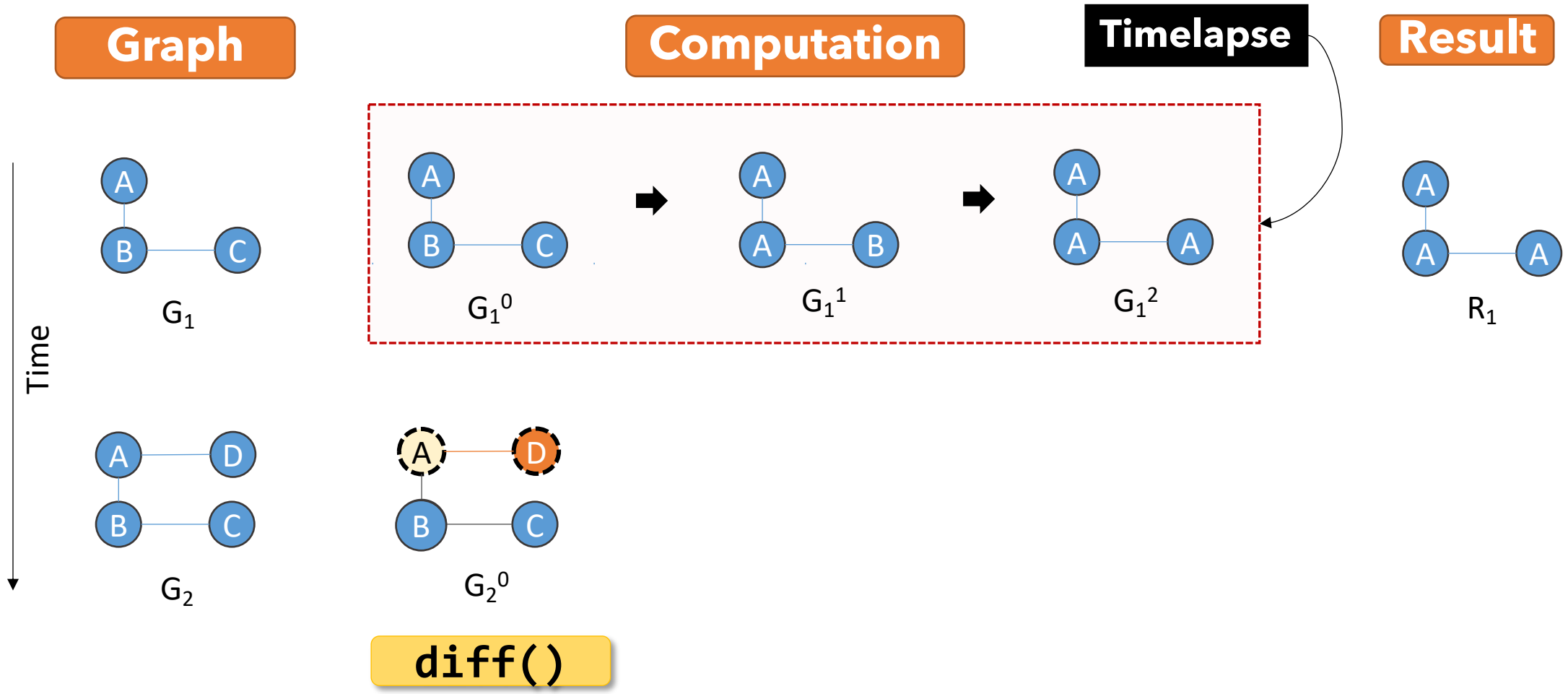
Computation Model

Leverages the characteristics of graph-parallel execution model



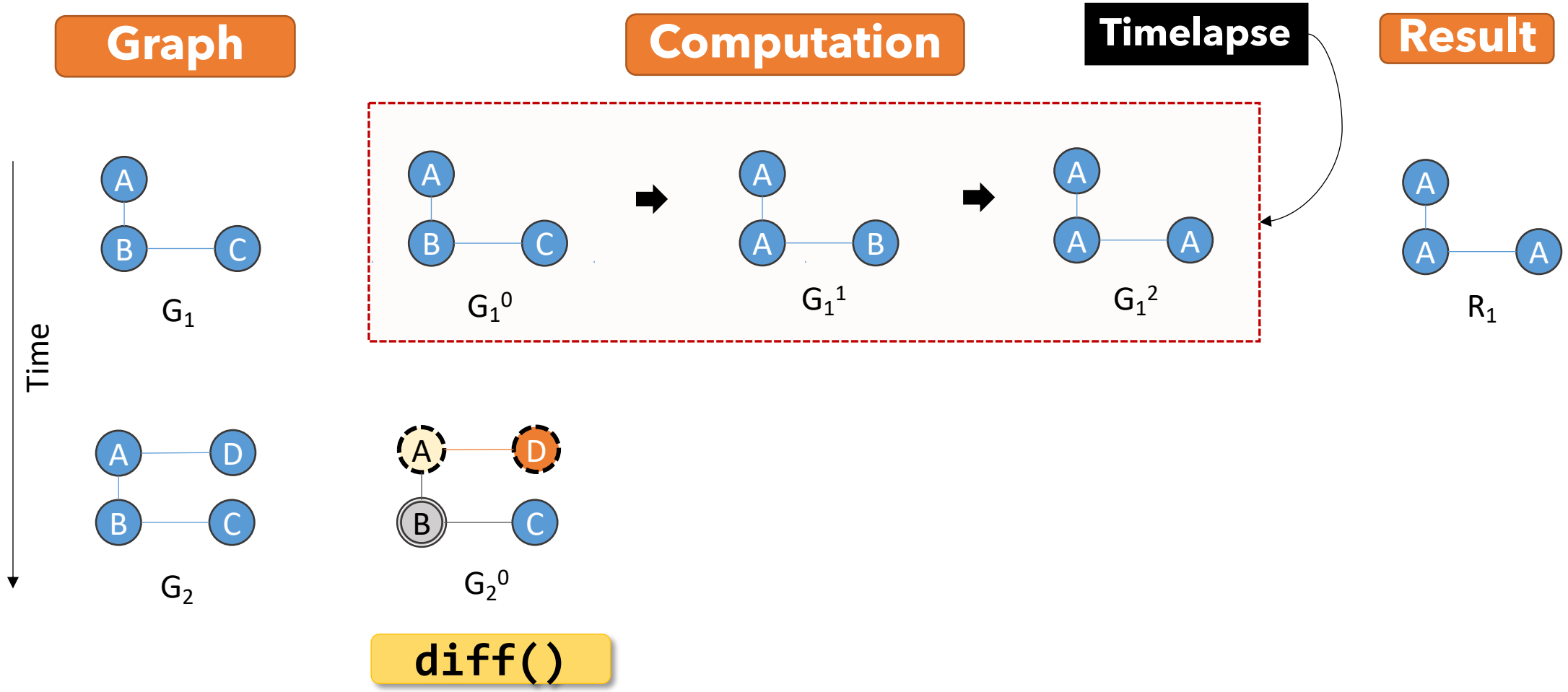
Computation Model

Leverages the characteristics of graph-parallel execution model



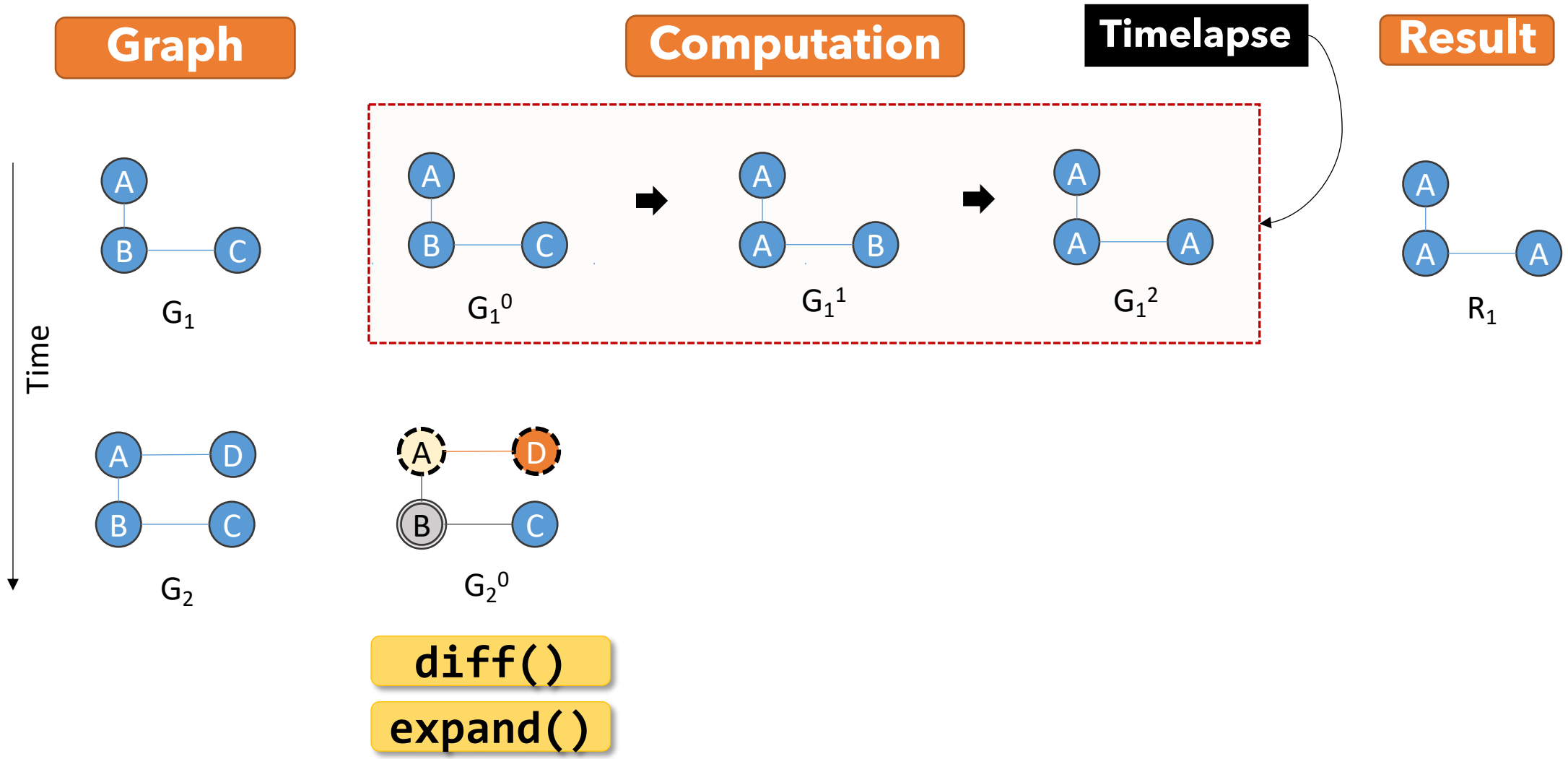
Computation Model

Leverages the characteristics of graph-parallel execution model



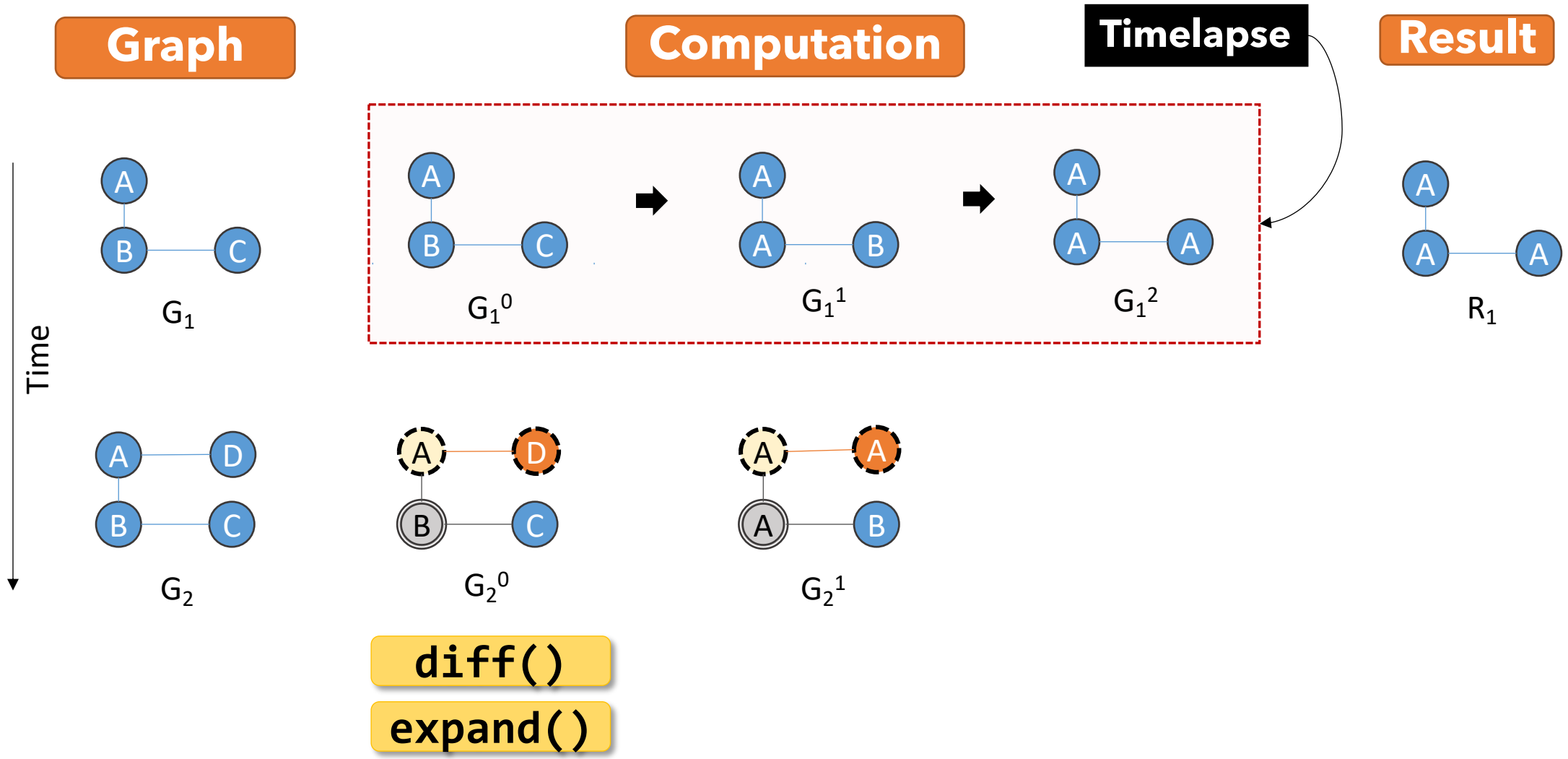
Computation Model

Leverages the characteristics of graph-parallel execution model



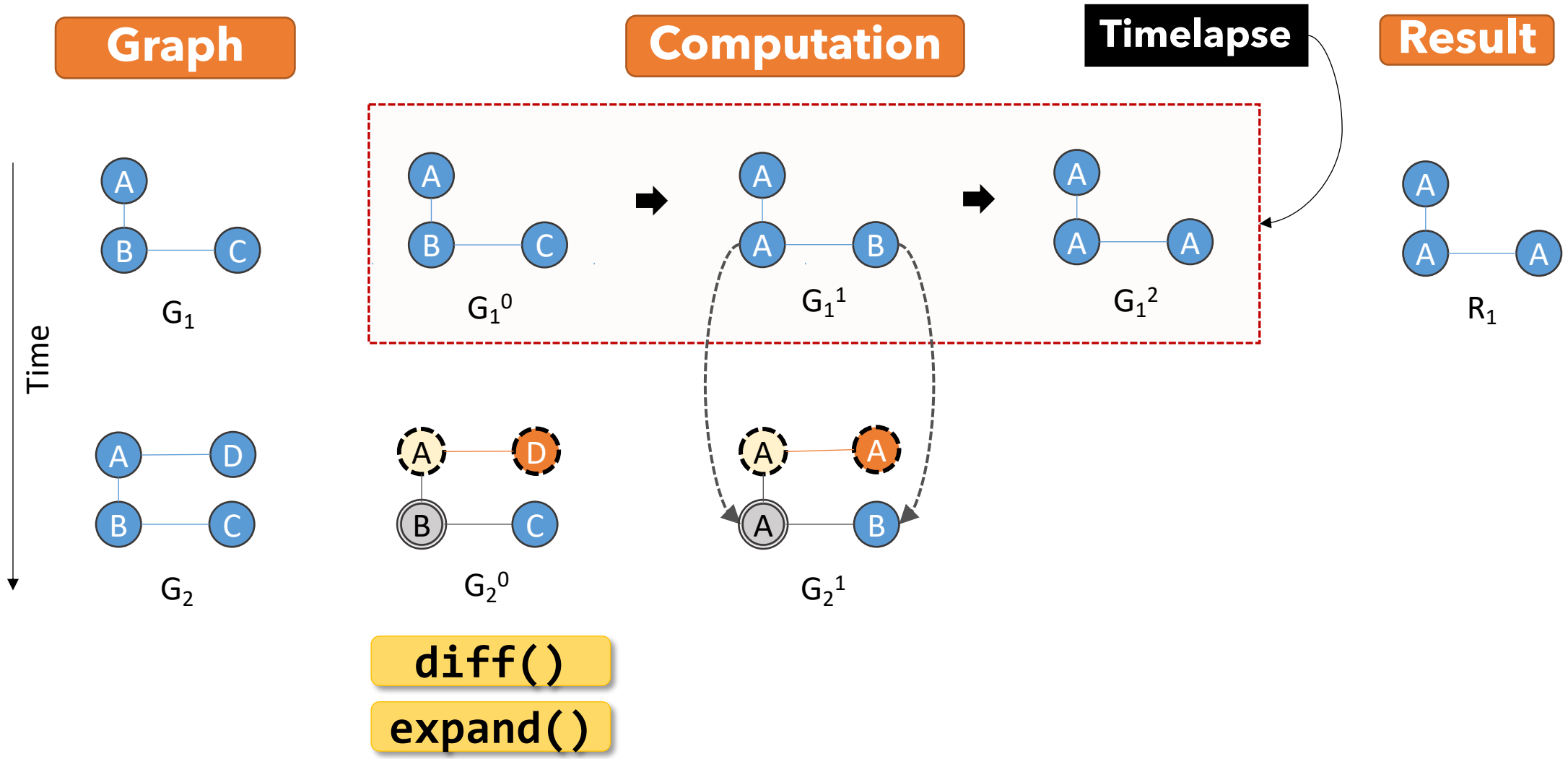
Computation Model

Leverages the characteristics of graph-parallel execution model



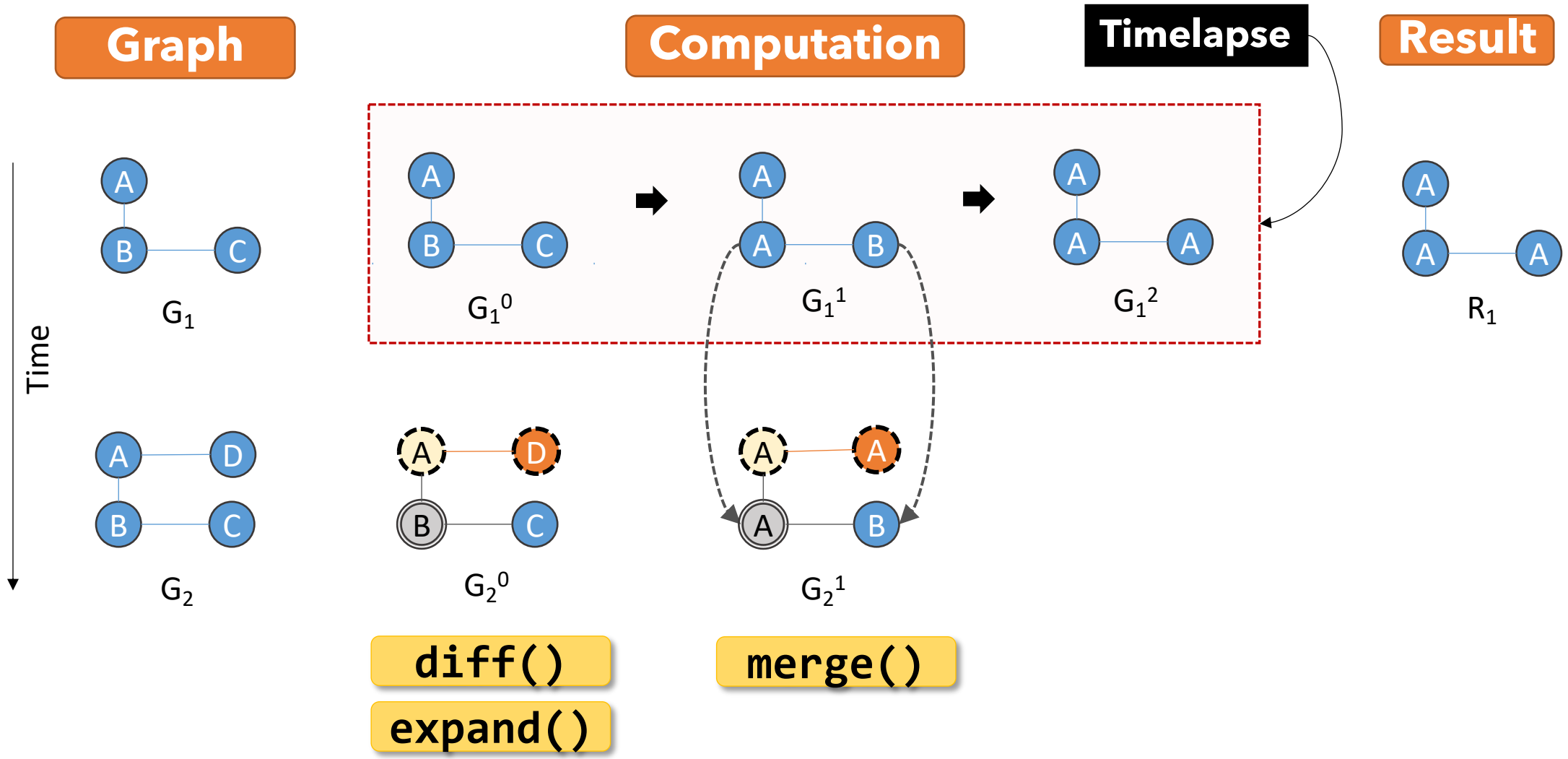
Computation Model

Leverages the characteristics of graph-parallel execution model



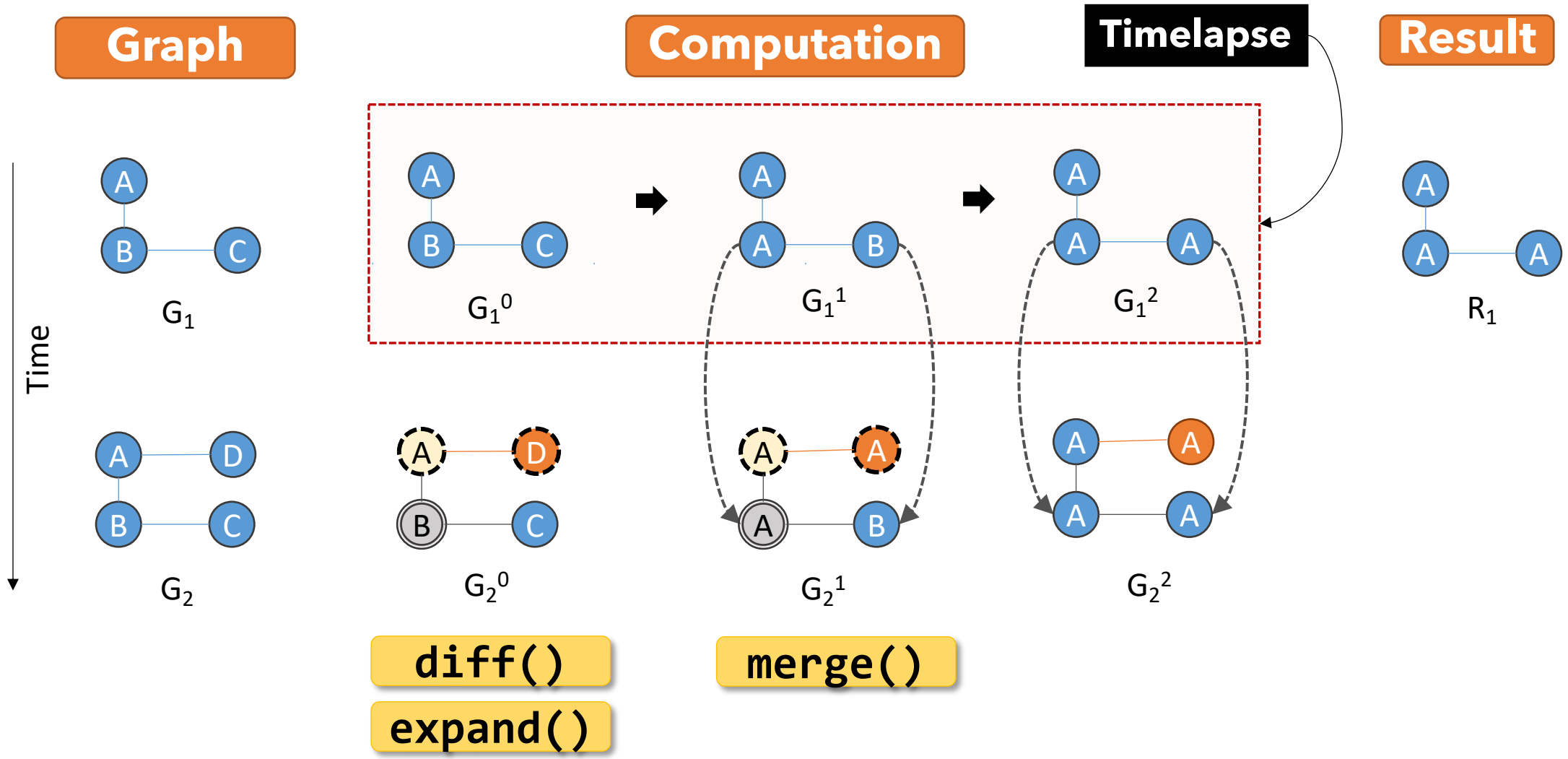
Computation Model

Leverages the characteristics of graph-parallel execution model



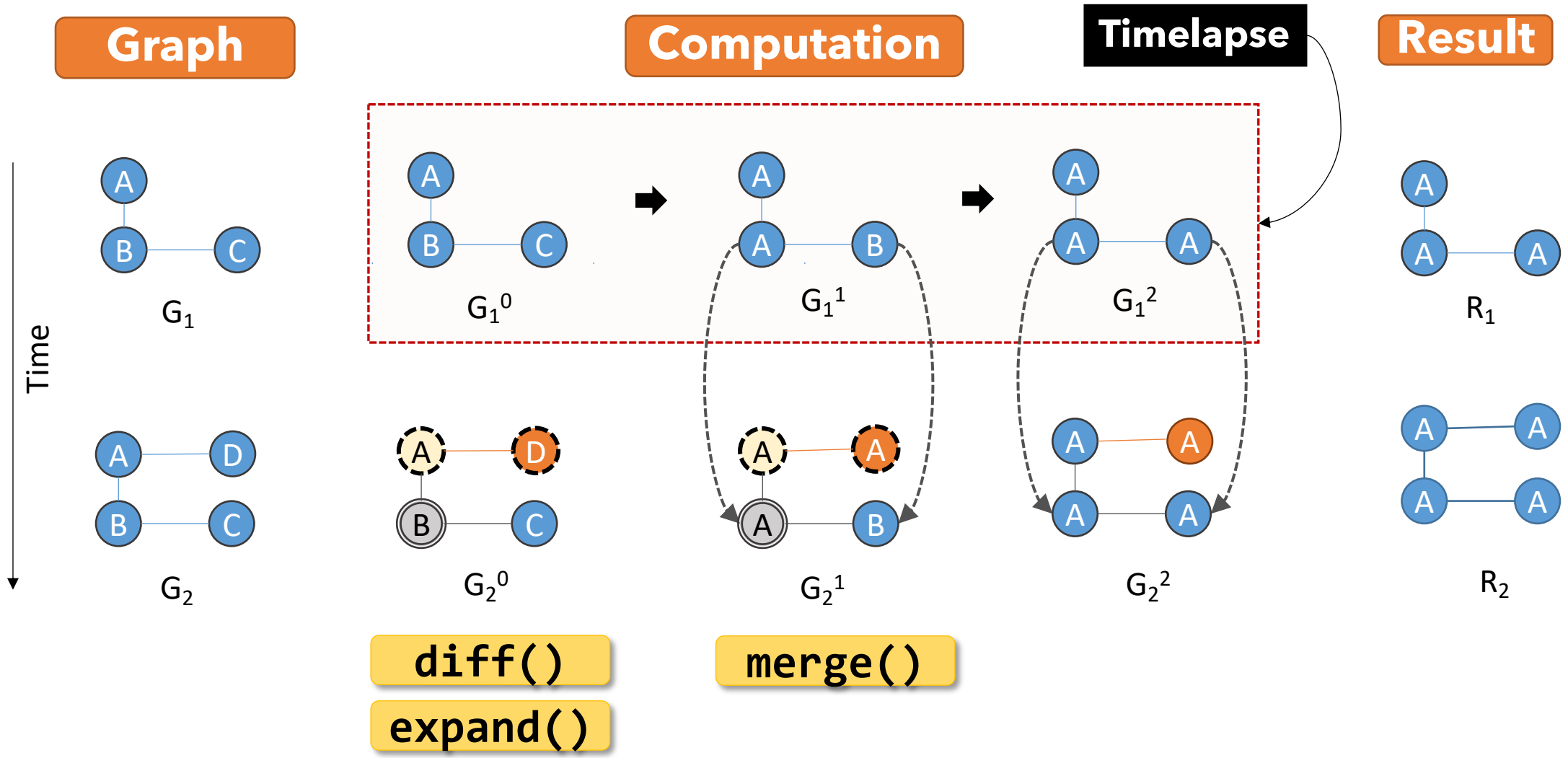
Computation Model

Leverages the characteristics of graph-parallel execution model



Computation Model

Leverages the characteristics of graph-parallel execution model



Computation Model

- Generates the **same** **intermediate state** for all entities at every iteration as compared to full re-execution
  - Correctness is preserved for any algorithm
- Computation **decoupled** **from state**
  - Can use any state, not just the previous
  - Can share state across queries
- Incremental computations **not** **always useful**
  - Can switch between incremental and full execution at any iteration
  - Uses random forest based model to do so

Details in the paper!

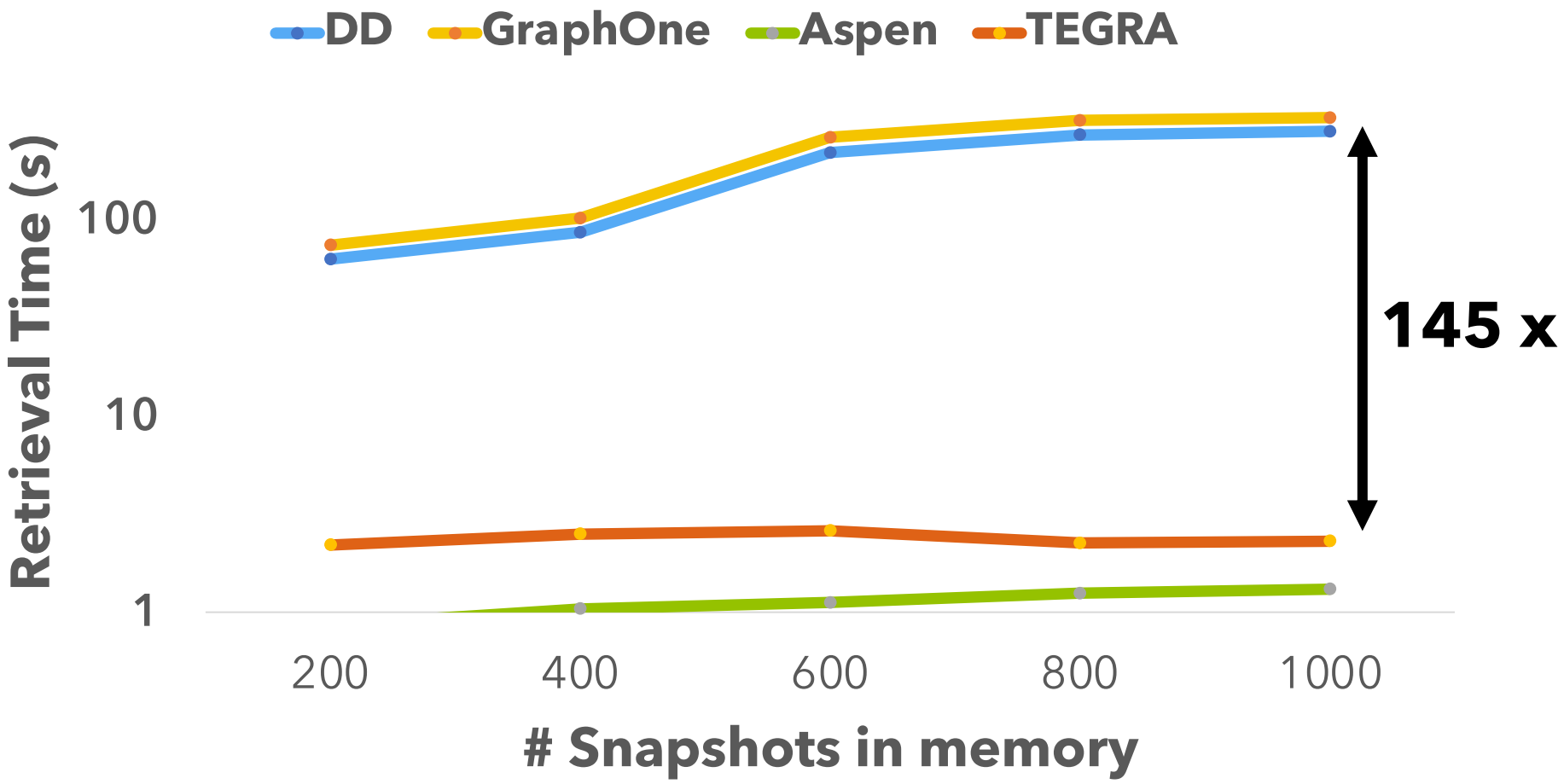
Implementation & **Evaluation**

- Implemented on Apache Spark
  - Drop-in replacement for GraphX
- Evaluated in a 16 node cluster
  - **Twitter:** 1.47B edges
  - **UK:** 3.73B edges
  - **Facebook:** 5B, 10B, 50B edges
- Comparisons & Algorithms
  - Differential dataflow [SOSP'13], GraphBolt [Eurosys '19], Chlonos [Chronos, Eurosys '14], Aspen [PLDI '19], GraphOne [FAST '19]
  - Connected components, Page rank, Belief propagation, and others.

More evaluation in the paper

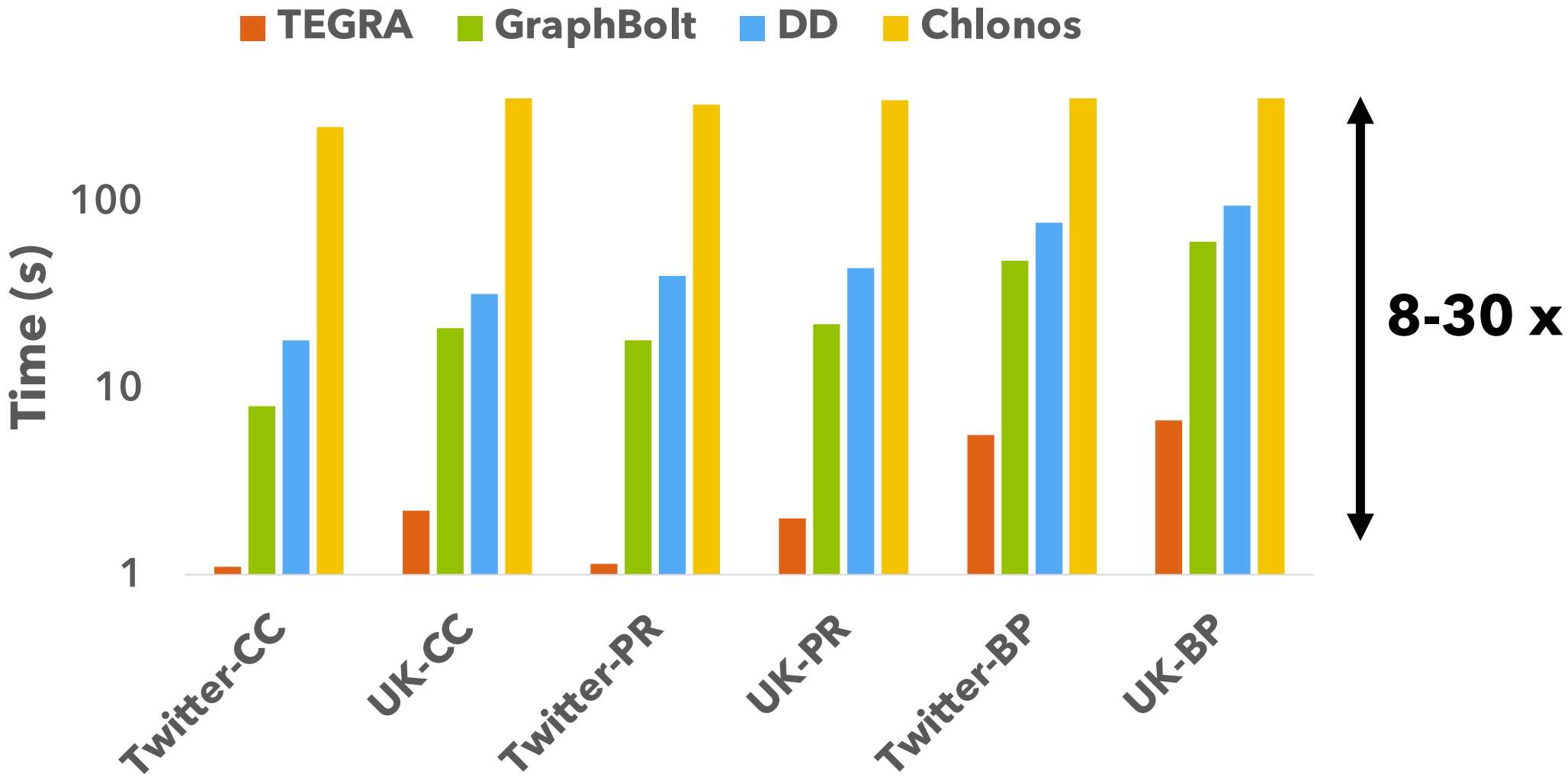
Ad-hoc **Retrieval**

Retrieval latency with varying amount of state in memory



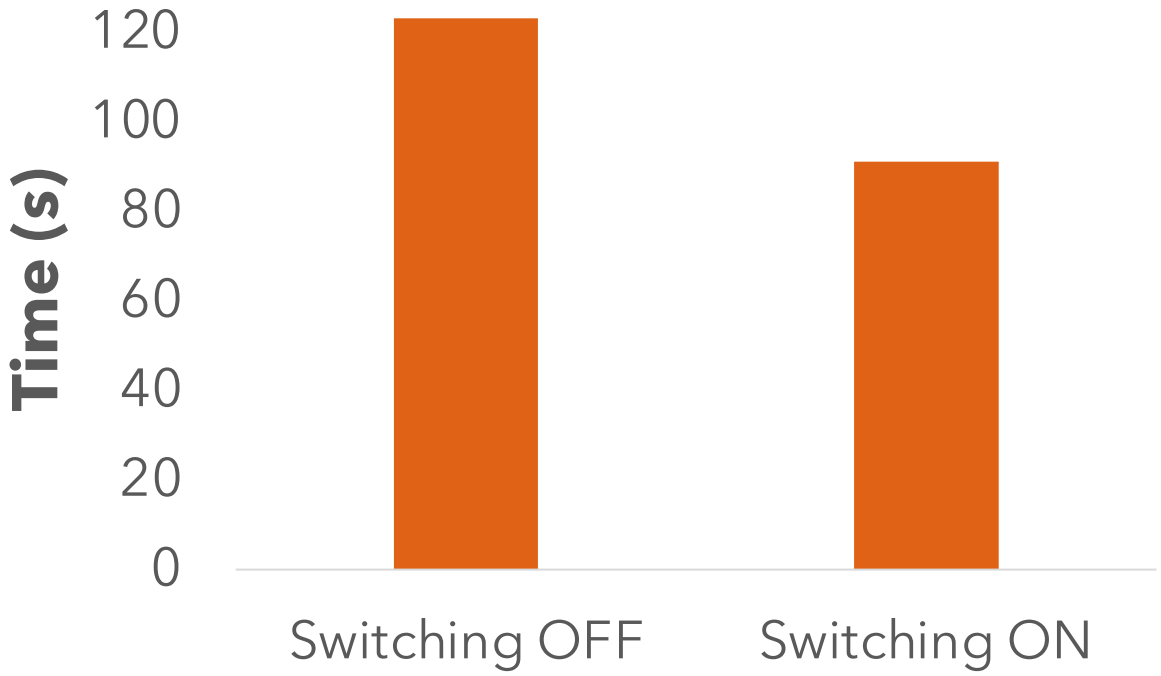
Ad-hoc **Analytics**

Random snapshot, use previous state if available

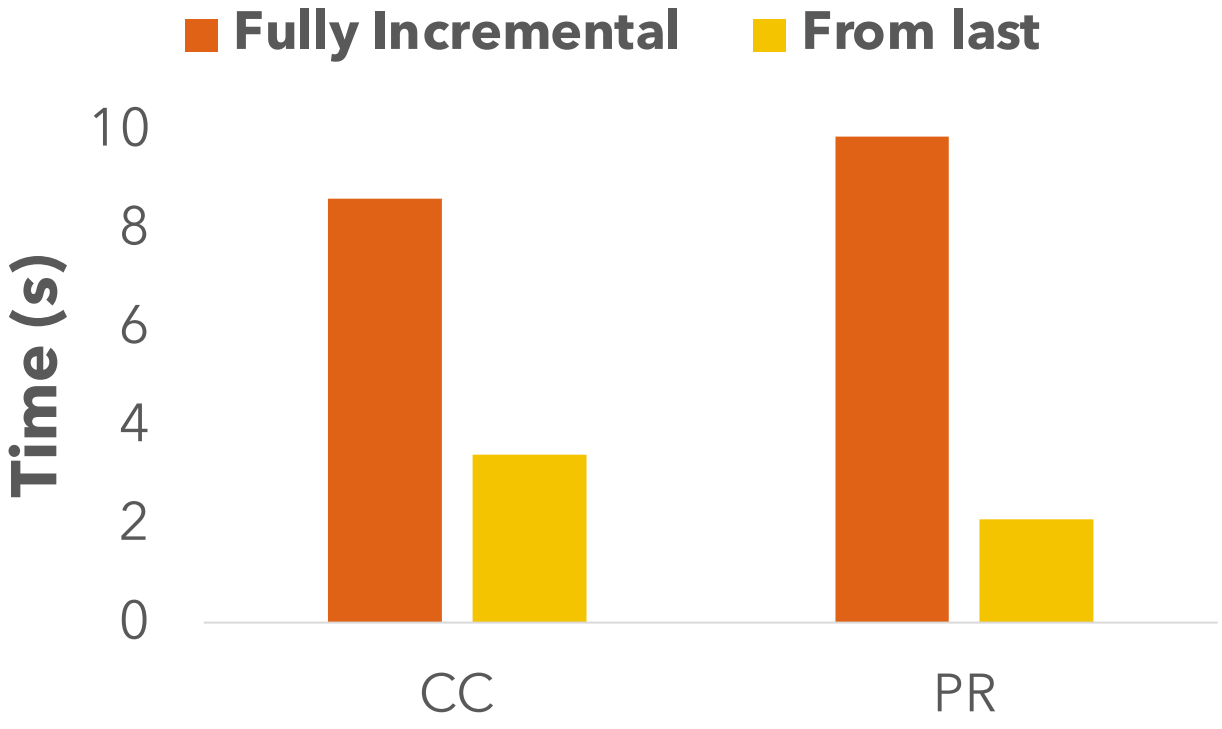


Computation Model Benchmarks

Switch to full re-execution



Flexibility in state reuse



Summary

- Ad-hoc analytics important emerging graph workload
- TEGRA enables efficient ad-hoc analytics on evolving graphs
  - Simple abstraction
  - Compact representation of state
  - Share storage and computation
- TEGRA outperforms state-of-the-art solutions

http://www.anand-iyer.com
anand.iyer@berkeley.edu