DREAM:Lab

From “Think Like a Vertex” to “Think Like an Interval”

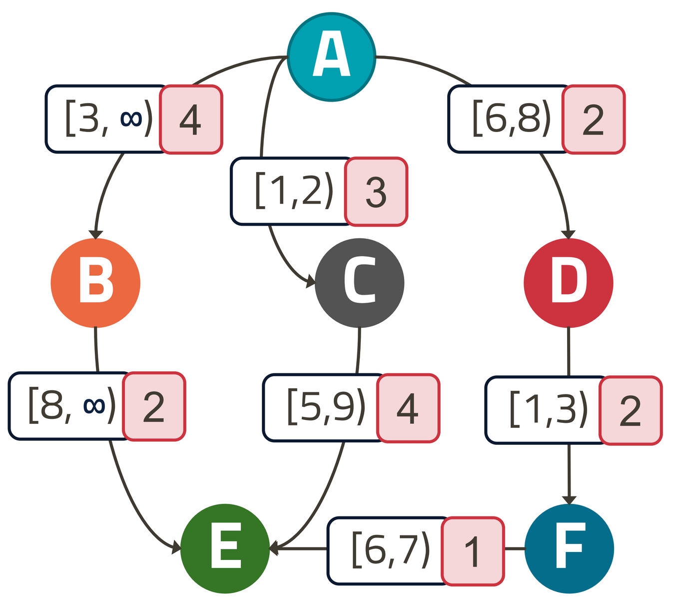
Swapnil Gandhi and Yogesh Simmhan

Introduction

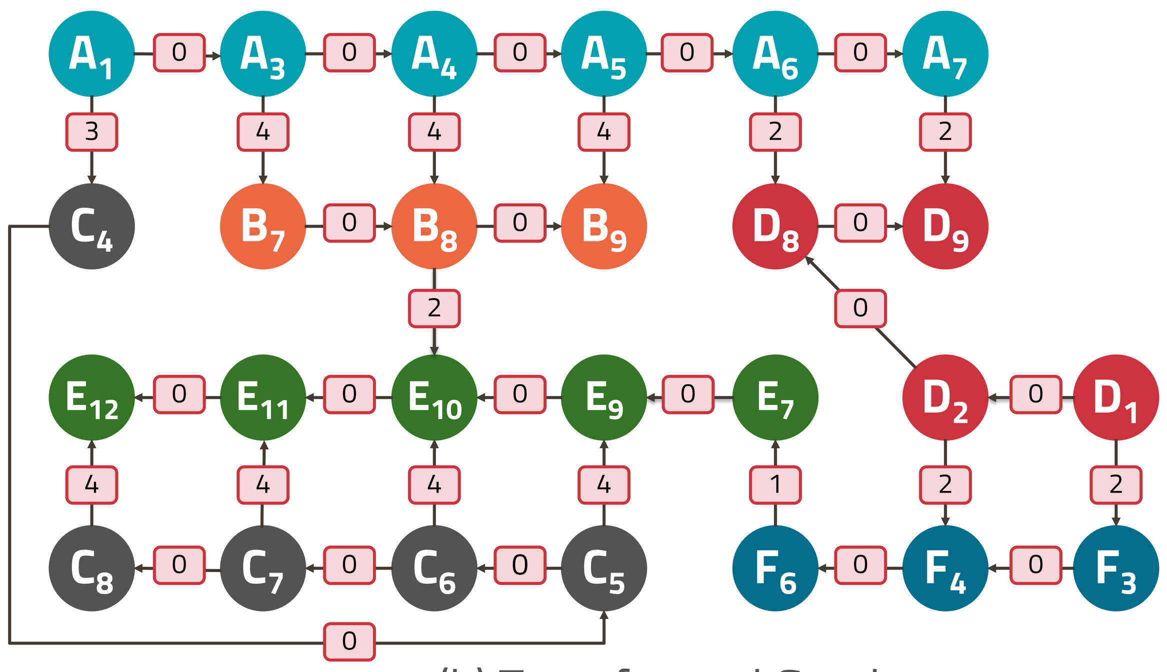
- Graphs are widely considered to be the natural means of representation for many static networks
- But real-world networks are often evolving with links being added and removed over time
- Temporal Graphs are dynamic networks which contain existence information for all vertices & edges at every point in time

  Examples : Social, Citation/Collaboration, Sensor & Transit Network, Human Connectomes, Internet-of-Things …

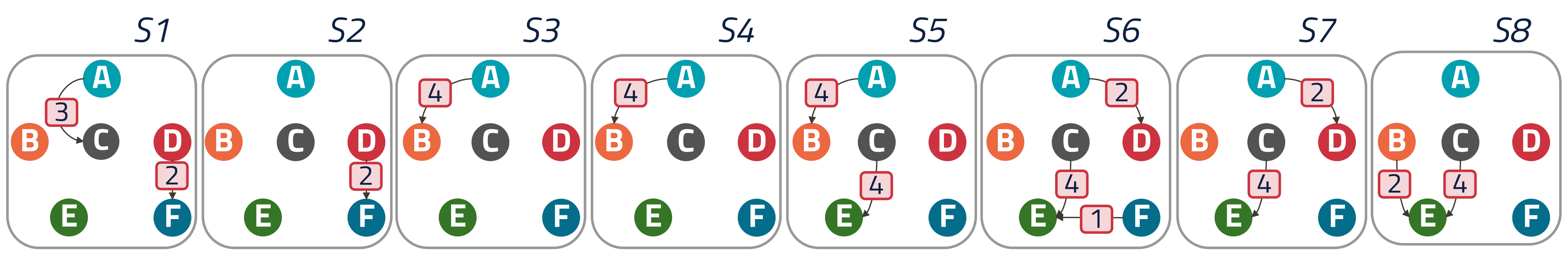
- Current de-facto representation for temporal graphs is a snapshot sequence, where state of graph is associated to a time-point



(a) Temporal Graph



(b) Transformed Graph



(c) Multi-Snapshot Graph

- Challenges:

1. Lack of *unifying abstraction* to operate on temporal graphs

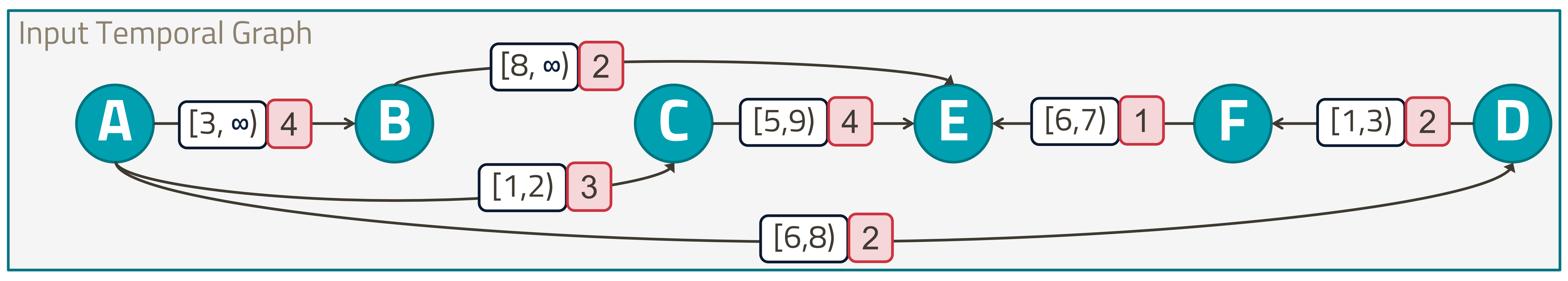
   Existing abstractions either work only for time-dependent algorithms or time-agnostic algorithms but not for both classes

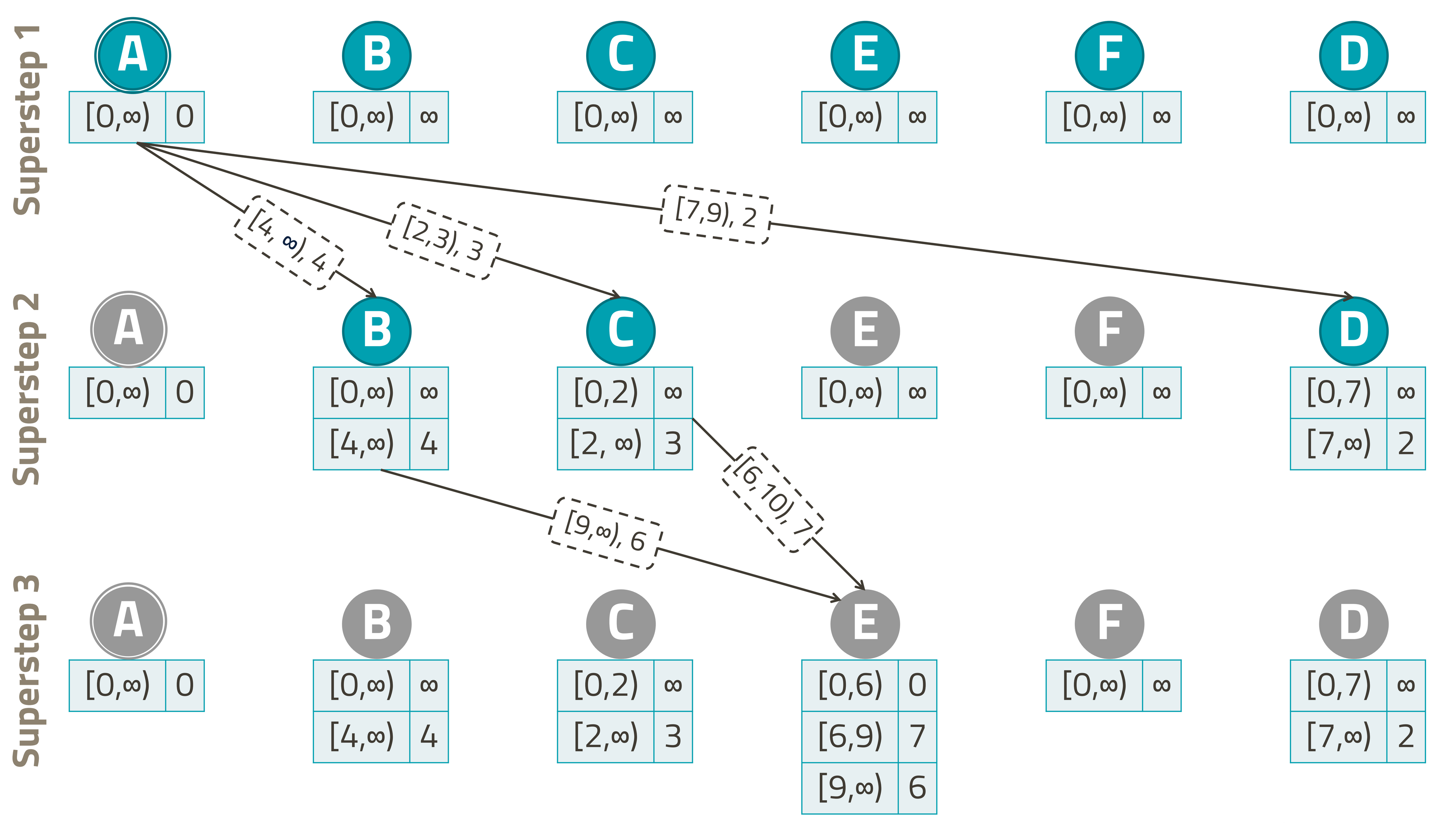
2. Existing abstractions either *do not scale* on distributed systems or are *not even designed* for distributed execution

   Specialized native time-dependent algorithms are designed for single-threaded shared-memory execution

This Work : *First scalable, distributed and fault-tolerant general-purpose programming abstraction for processing arbitrarily large temporal graphs*

Single-Source Shortest Travel Time

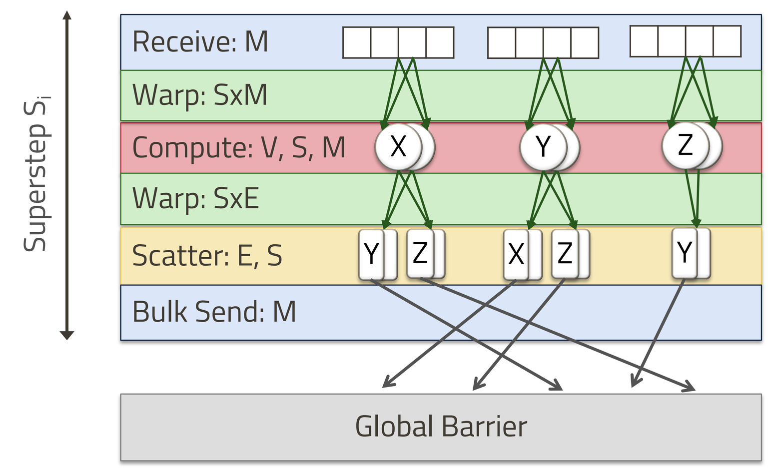




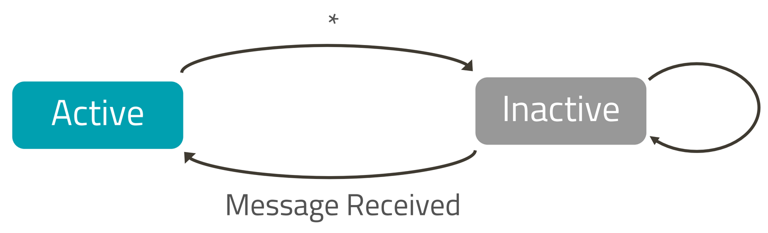
*Travel-Time : 1 Unit

Interval-Centric Computation

- ICM extends vertex-centric computation model with Interval as a first-class citizen
  - Vertex (and edge) attributes, computation state and messages have *time-validity* and may have associated computation



(d) Supersteps in ICM



(e) Interval-vertex State Machine

- User writes program from the perspective of a single interval
  - In each superstep, user program invoked once for each active interval
  - Communication via pure message passing using BSP
  - Synchronicity avoids dead-locks and data-races
- In superstep S_i, user program can read messages send in superstep S_{i-1}, modify current computation state, and send new messages to other vertex intervals

Time-Warp

- Allows user program to consistently operate over temporal messages and partitioned computation state
  - Reminiscent of Temporal Aggregate
  - Uses one-pass algorithm with support for online aggregation

State

τ_s	S
[0,5)	s_1
[5,9)	s_2
[9,10)	s_3

Messages

τ_m	M
[0,4)	m_1
[2,7)	m_2
[5,7)	m_3
[5,9)	m_4
[9,10)	m_5

Time Warp $\mathbb{X}_{S\times M}$

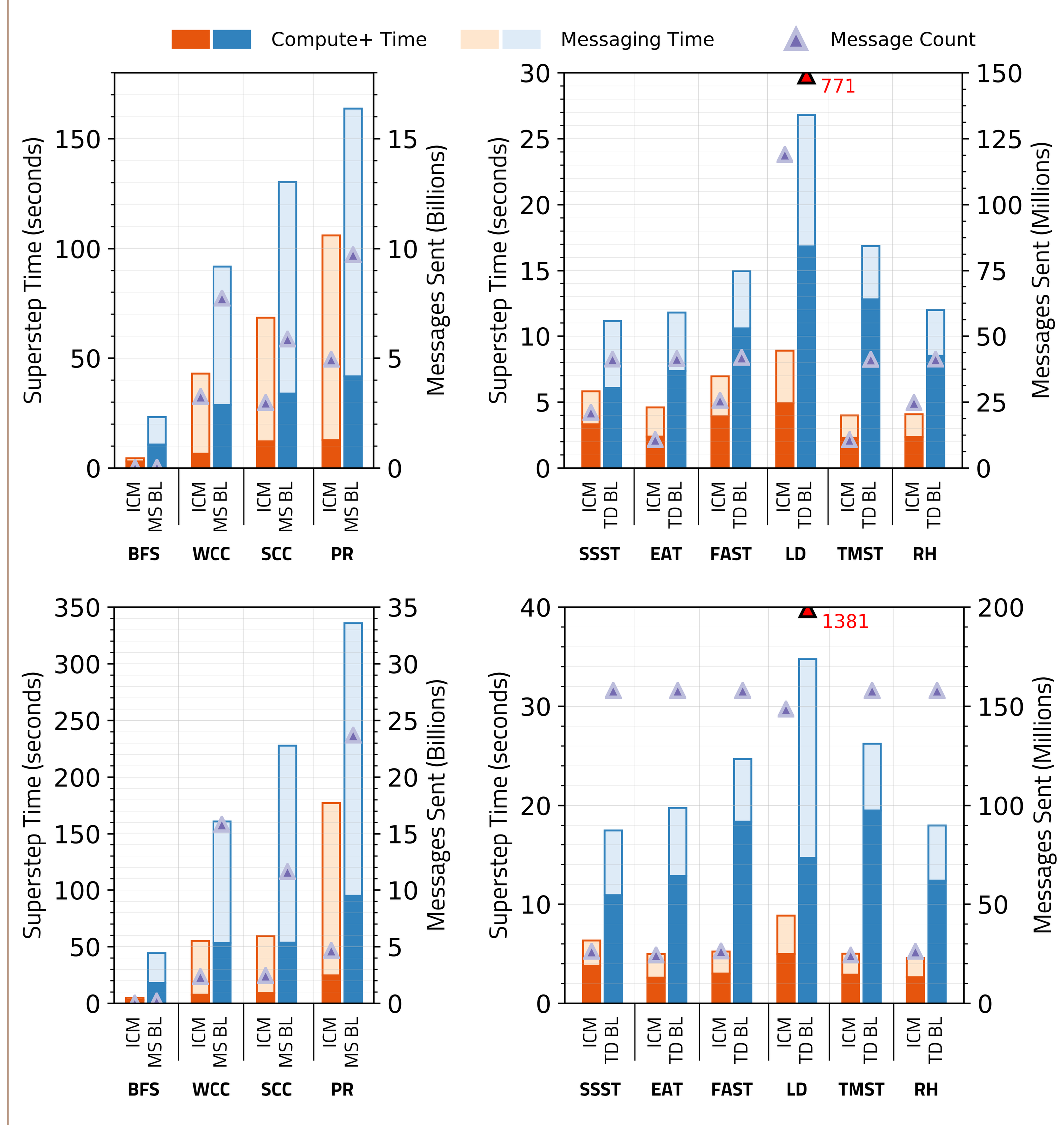
TW	τ	S	$\mathbb{M}$
w_1	[0,2)	s_1	m_1
w_2	[2,4)	s_1	m_1, m_2
w_3	[4,5)	s_1	m_2
w_4	[5,7)	s_2	m_2, m_3, m_4
w_5	[7,9)	s_2	m_4
w_6	[9,10)	s_3	m_5

Advantages :

- Guarantees minimal number of user program invocation
- Transparently coalesces interval-state and outgoing messages

Experimental Evaluation

Dataset	Snapshots	Median Lifespan V	Median Lifespan E	ICM Graph $\lvert V_{ICM}\rvert$	ICM Graph $\lvert E_{ICM}\rvert$	Multi-Snapshot Graph $\sum\lvert V\rvert$	Multi-Snapshot Graph $\sum\lvert E\rvert$	Transformed Graph $\lvert V_{TD}\rvert$	Transformed Graph $\lvert E_{TD}\rvert$
LDBC G1	10	4	6	8.9M	251M	45M	1.3B	58M	1.4B
LDBC G2	10	5	9	9M	260M	81M	2.4B	82M	2.4B



1.5x-6x faster runtime
2x-7x Reduction in #Messages Sent
~4x Reduction in #Compute Calls

2x-10x faster runtime
2x-11x Reduction in #Messages Sent
~7x Reduction in #Compute Calls

Longer Vertex and edge lifespan ➲ Better performance benefits

Summary & On-Going Work

- First scalable API for processing temporal graph
  - Leverages existing solutions & makes it practical
- On-going work towards adding support for real-time streaming updates and temporal graph partitioning

Distributed Research on Emerging Applications & Machines Lab
Department of Computational and Data Sciences
dream-lab.in | Indian Institute of Science, Bangalore

DREAM:Lab