VAST • INFOVIS • SCIVIS
BIOVIS • LDAV
VIS 2013

Visualizing request-flow comparison to aid performance diagnosis in distributed systems

Carnegie Mellon University

Raja Sambasivan,
Ilari Shafer, Michelle Mazurek, Greg Ganger

Distributed systems' traits & needs

- Used by society for many purposes

- Prone to difficult-to-solve problems
  - Due to scale & complexity
- **Key need**: Sophisticated diagnosis tools

Diagnosis technique/tool research

- Very active research area
  - E.g., Distalyzer, NetMedic, Spectroscope...

- Most only automatically localize problem
  - Developer must use results to ID root cause

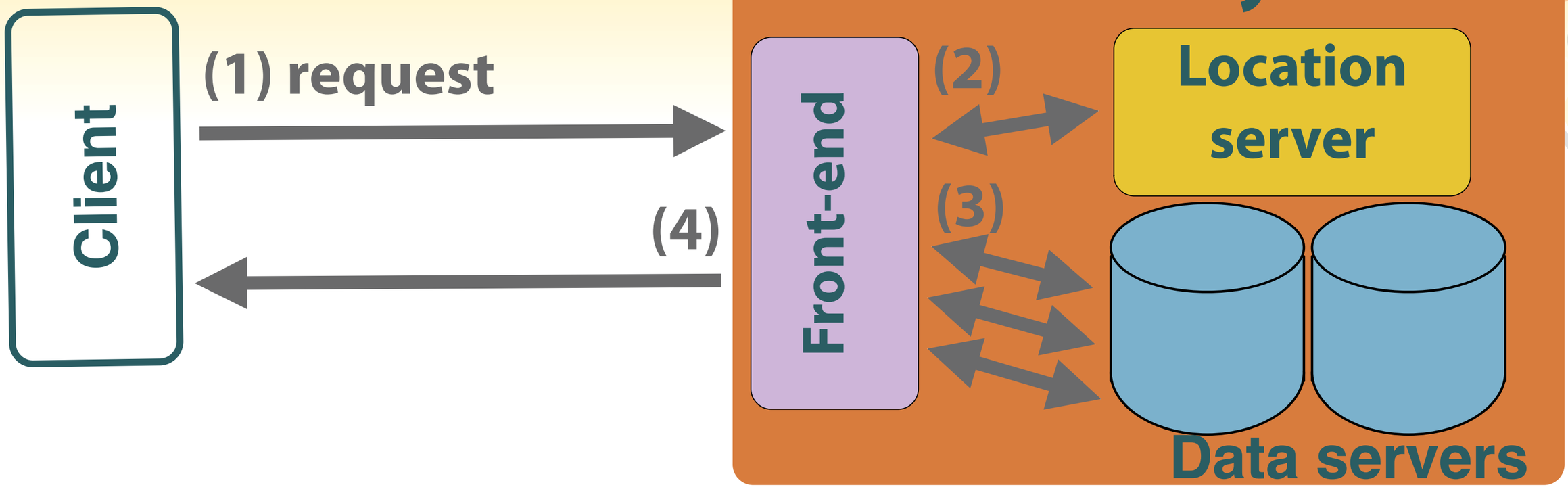
Tools must visualize results well to be effective

Visualizing diagnosis tools' results

- Little guidance on what what works best for **distributed system developers/problems**
  - There are a few exceptions
- Limits utility of diagnosis tools

This talk: User studies to find good approaches for request-flow comparison

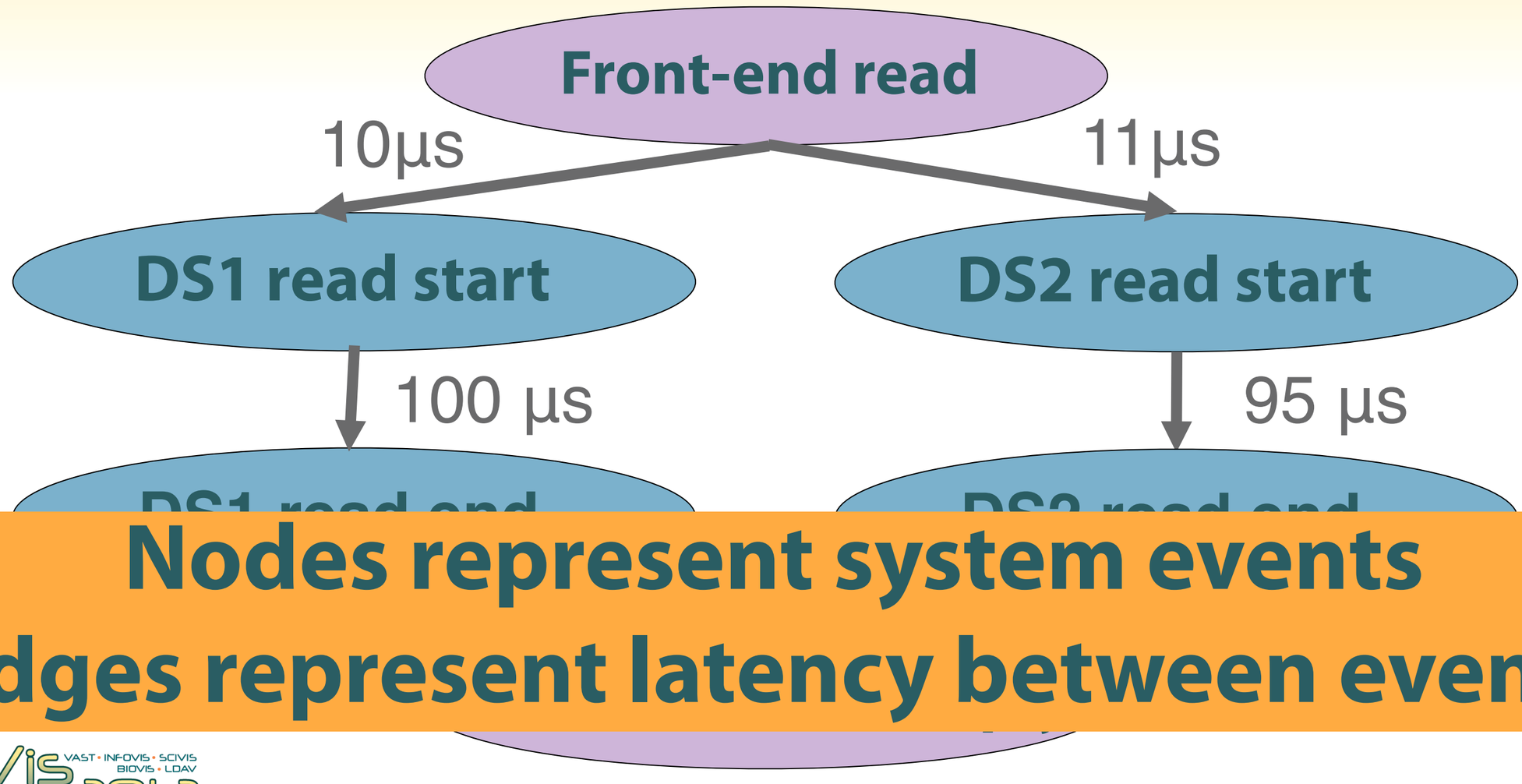
Example distributed system



- Work done to satisfy request can be represented as a workflow (i.e., a DAG)

An example request flow

Response time: 1,090µs



Nodes represent system events
Edges represent latency between events

Spectroscope [NSDI'11]

- Localizes performance degradations
  - By ID'ing changed request flows

- Output:
  - Groups of before/after request flows
  - Some changes automatically ID'd

Goals of visualization

- For a given before/after DAG in the output:
  - Help developers understand problem
  - Show automatically ID'd timing changes
- Help identify important timing changes that weren't automatically identified
- Help identify changed substructures

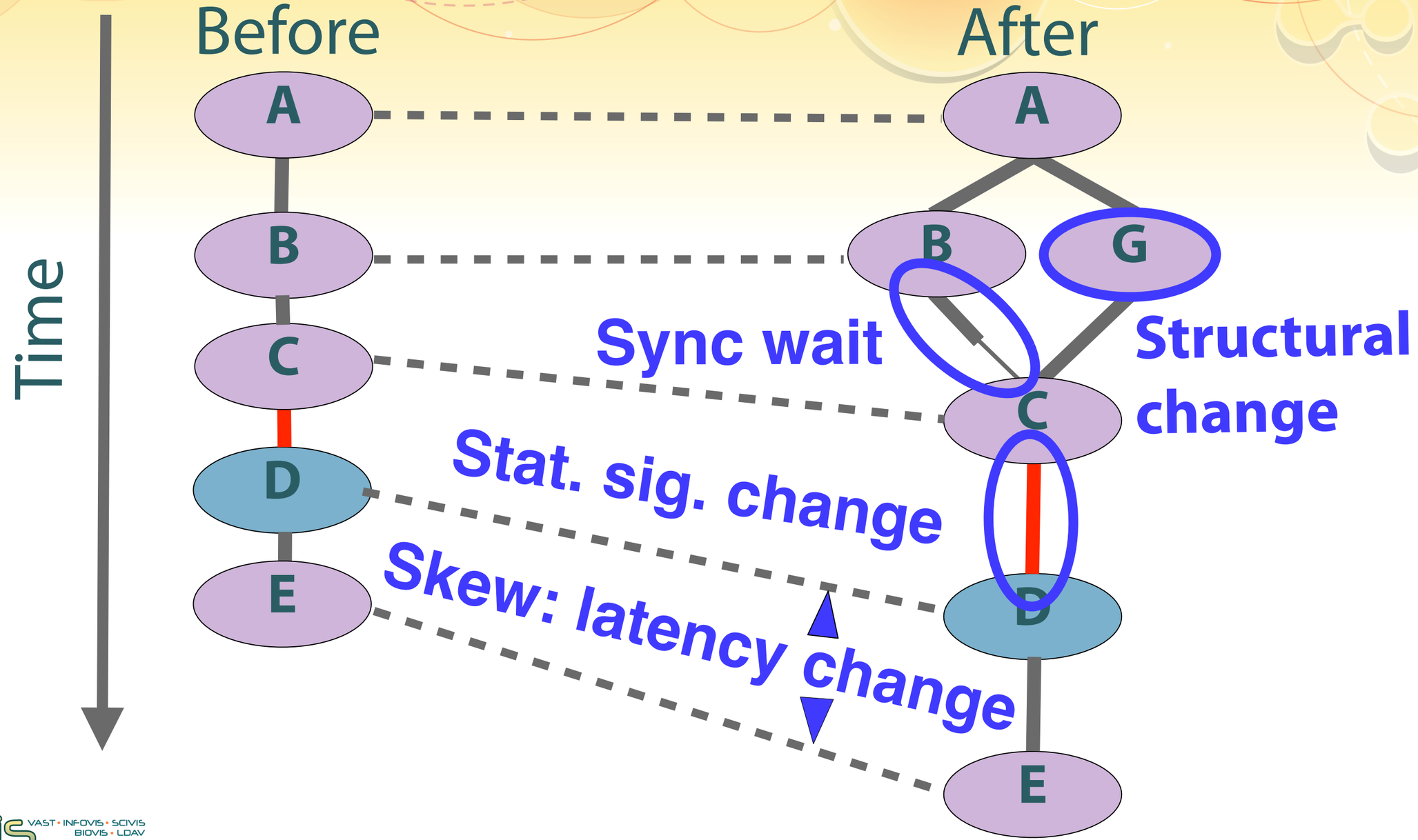
Side-by-side, diff, animation

- Suggested by previous Spectroscope users who used a more basic presentation layer

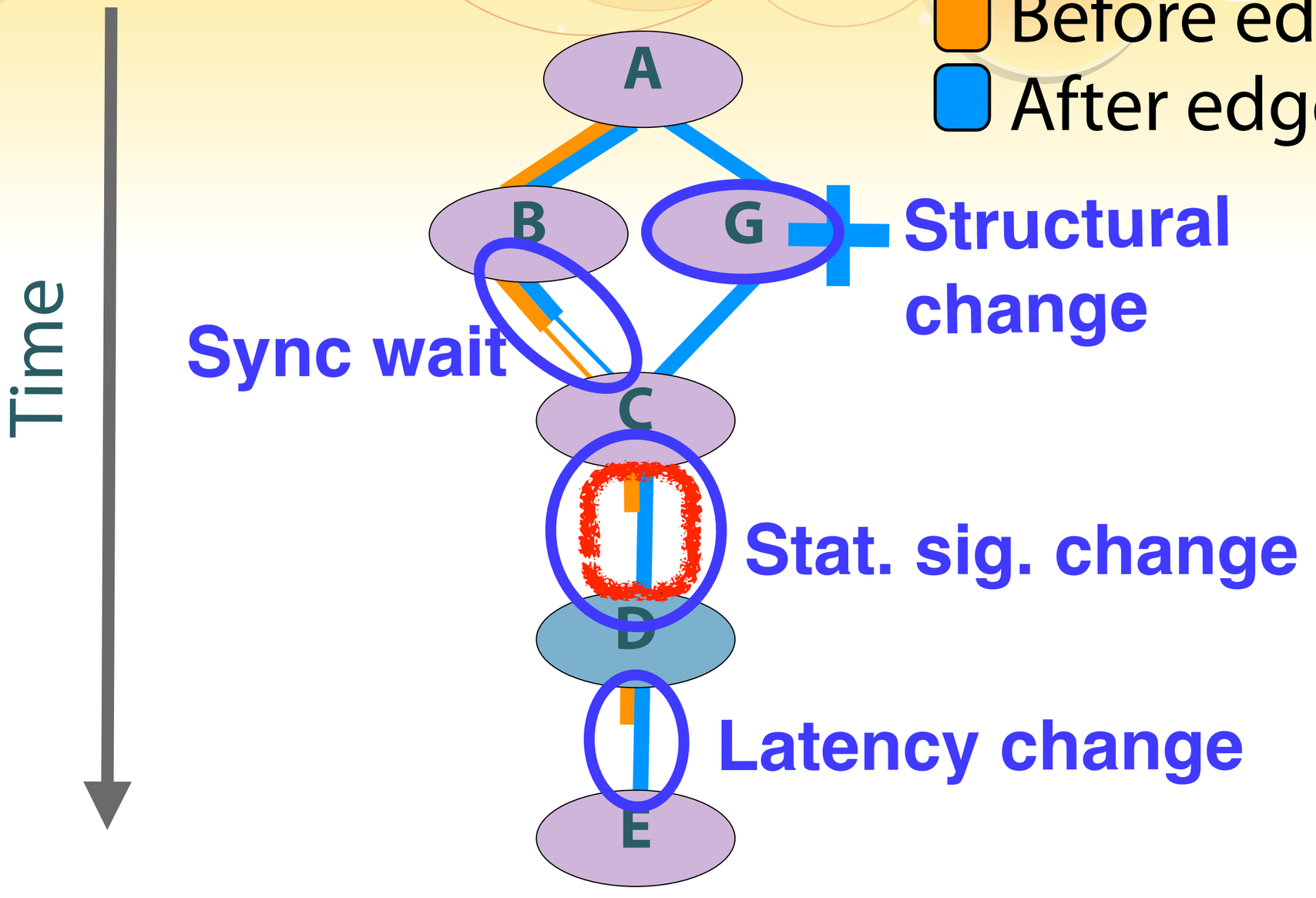
- Represent orthogonal approaches

- Well-studied in different contexts

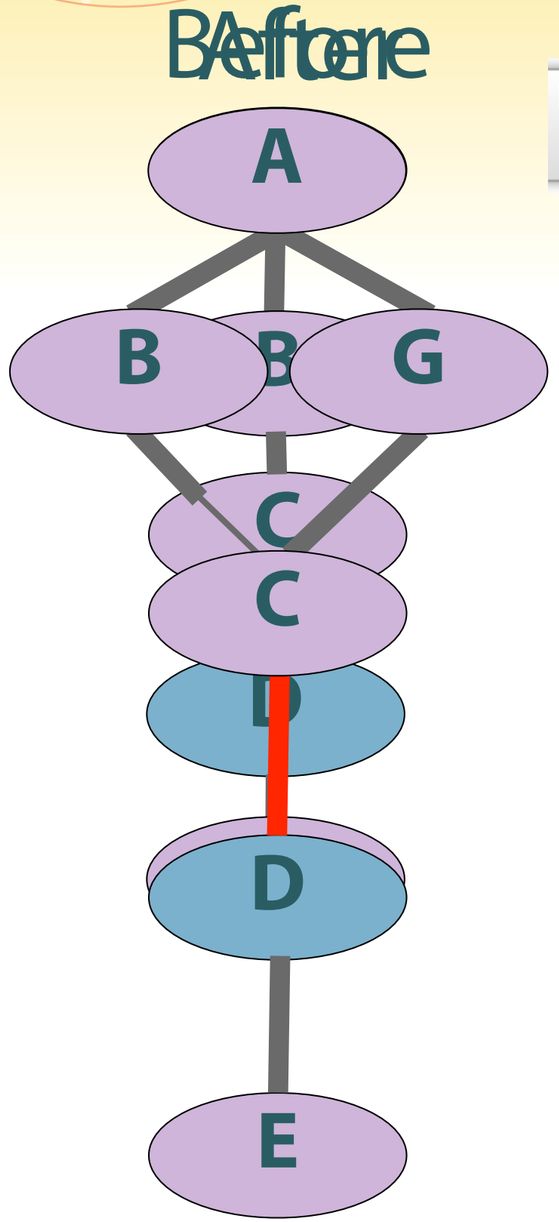
Side-by-side implementation

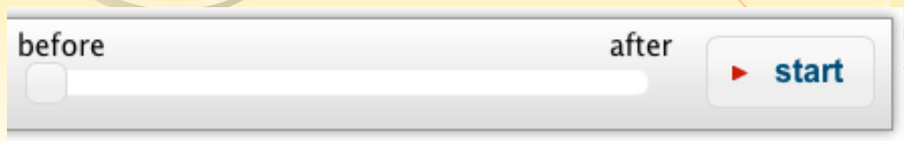


Diff implementation



Animation implementation





User study overview & methodology

- Deployed a between subjects study
  - Composed of 26 participants
- Users diagnosed 5 Ursa Minor problems
  - Users asked to complete 5 tasks for each
  - Collected completion times & comments

Overview of results

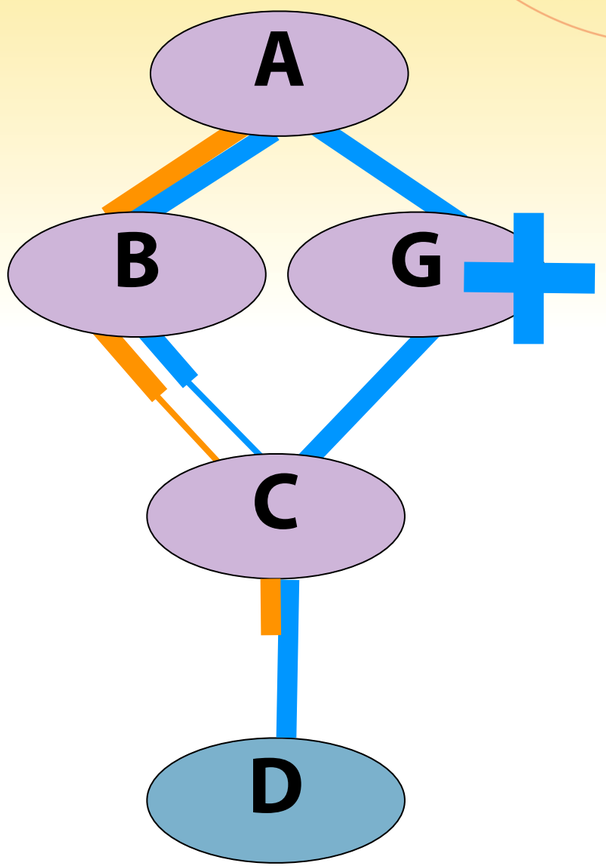
- No clear winner
  - If I had to choose...I'd be sad. (PD06)
- Animation approach slower for most tasks
- Different approaches best for diff. problems

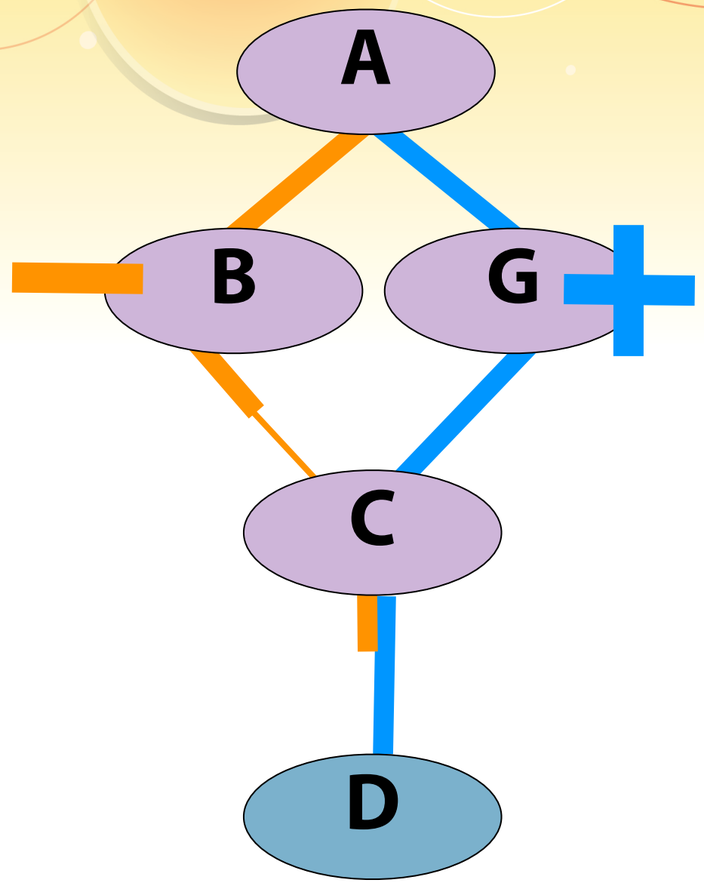
Best approach for problem types (I)

	Structural			
Approach	Edge latency	Concurrency	Intra-thread	Intra-thread mixed
Side Diff	✔			

Diff: concurrency vs. intra-thread change

Extra concurrency





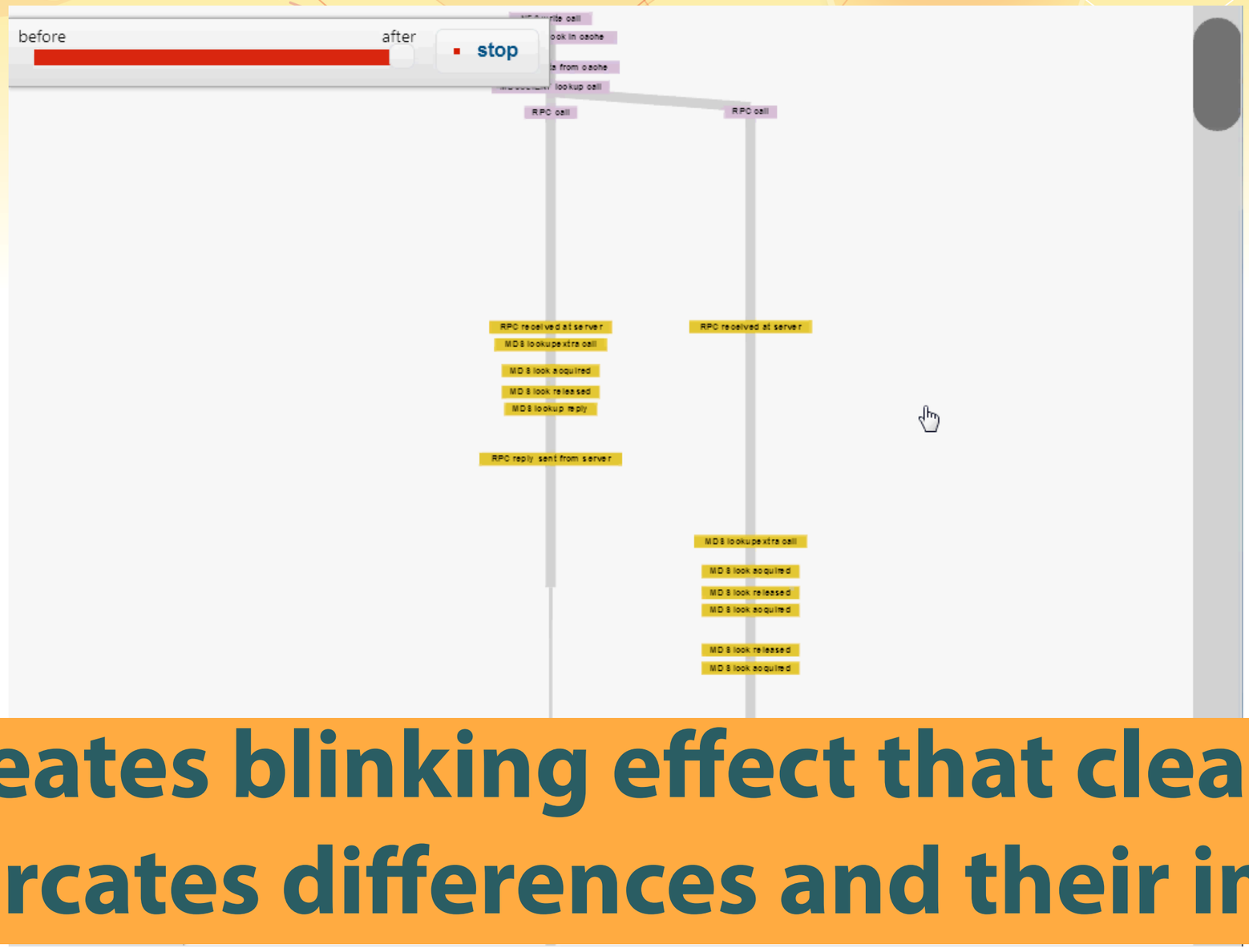
Intra-thread addition

Similar representation
makes it easy to confuse one for the other

Best approach for problem types (II)

		Structural		
Approach	Edge latency	Concurrency	Intra-thread	Intra-thread mixed
Side Diff Anim	✔	✔	✔	✘

Anim: good for structural changes



Creates blinking effect that clearly demarcates differences and their impact

Summary

- Conducted user study to find approaches suited to visualizing req.-flow comparison
- Found no single best approach
- See paper for more results & design lessons