Space-Time Memory

a concurrent dynamic data structure for flexible manipulation of time-sequenced data, with automatic GC

Kishore Ramachandran

Georgia Tech

Joint work with researchers from **Compaq CRL**

(**Rishiyur Nikhil, Jim Rehg, Bert Halstead, Chris Joerg, Leonidas Kontothanassis** and **Kath Knobe**)

Interactive Stream-Oriented Apps

- vision, animation, multimedia collaboration

Why parallel computing for such apps?

- computationally intensive
- inherently parallel (pipelined, data, and task)

Platforms?

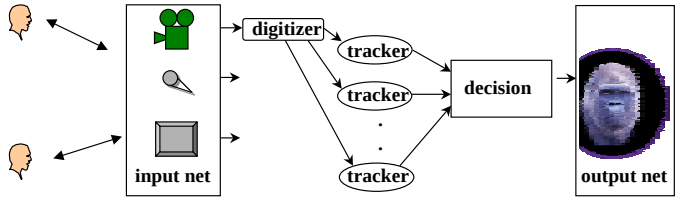
- SMPs, and clusters

Problems

- dynamic data sharing
- real-time properties

CRL's Smart Kiosk Application

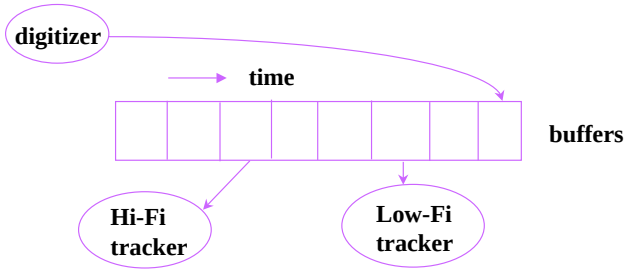
- public access to info and entertainment
- multiple users interact with multiple Kiosks
- input: implicit (camera, gaze, infrared,...) and explicit (voice, gesture, touch-screen,...)
- output: emotive face, synthesized speech, ...



heterogeneous pieces of software: GUIs, trackers, etc.

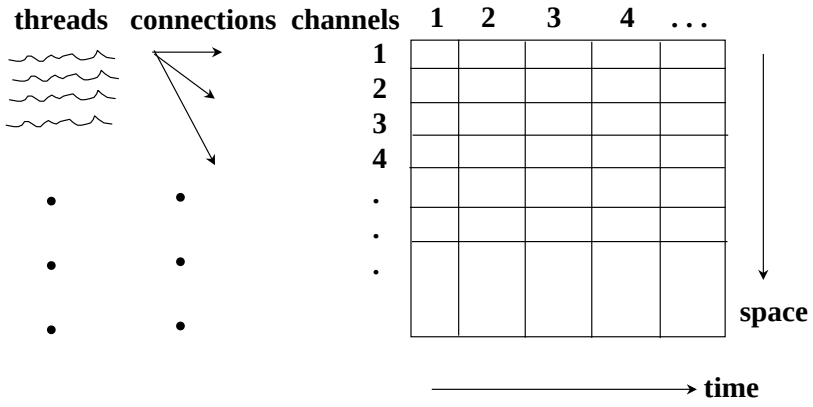
What new issues?

- temporally evolving dynamic data structures
- dynamic producer-consumer relationships
- not everything consumed
- inter-stream synchronization

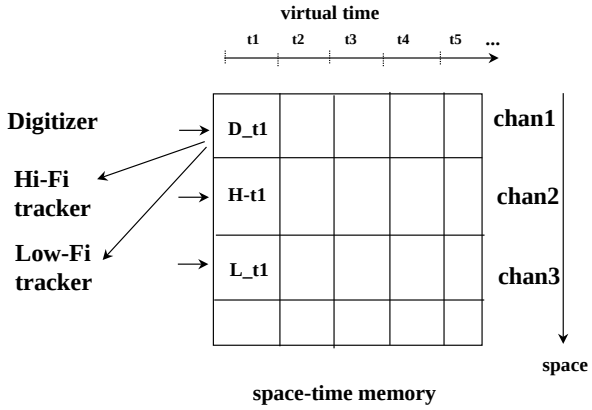


Space-Time Memory

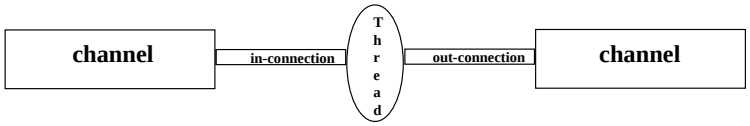
Concurrent dynamic data structure



An Example



Using the STM



consumer:
get-item(in-connection, ts)
code to use item
consume-item(in-connection, ts)

producer:
put-item(out-connection, ts)

API includes calls to:
create channel
connect, disconnect to/from channel
advance thread virtual time
synchronize virtual time with real time

Summary

- STM, a concurrent dynamic data structure for flexible manipulation of time-sequenced data, with automatic GC
- Why is STM a good idea?
  - time: important attribute for interactive apps
  - sharing abstractions such as DSM, and synchronization abstractions such as locks and barriers are too low level
  - current parallel programming languages do not offer the right abstractions for stream-oriented interactive apps.