# TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications


**Philip Levis**
UC Berkeley
Intel Berkeley

**Nelson Lee**
UC Berkeley
Stanford

**Matt Welsh**
Harvard University

**David Culler**
UC Berkeley
Intel Berkeley

# A Lesson from History

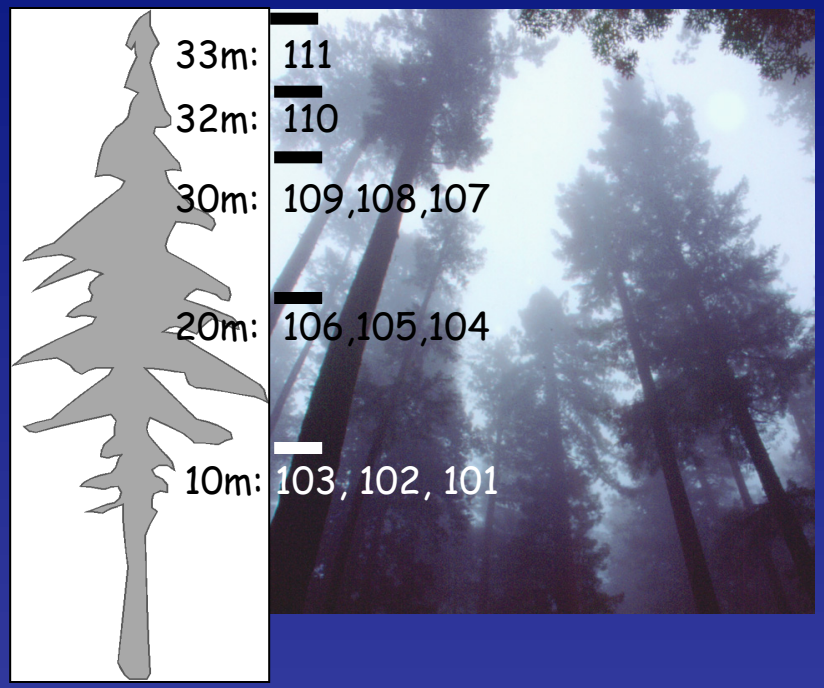
- Simulation can provide
  - The ability to inexpensively explore ideas
  - A controlled environment
  - Support for extensive observation
- Examples
  - SimOS
  - SimpleScalar
  - ns-2
- Accelerated research in their domains

# But…

**Capturing the essential elements of their domains was critical to the utility of these simulators.**

# Mote Sensor Network Domain?

- Many tiny devices
- Embedment
- Non-deterministic
- Application-specific
- System interactions



# What Do We Need to Capture?

- **Bridging**
  - **Implementations, not just algorithms**
- **Completeness**
  - **Spectrum of low-level protocols to applications**
  - **Applications adapt to their environment**
  - **Capture the cross-layer interactions**
- **Fidelity**
  - **Capture these interactions at a very fine grain**
- **Scalability**
  - **Examine behavior in dense or large networks**

# Our Approach: TOSSIM

- Simulator for TinyOS programs
- Compiles directly from TinyOS code
  - Executes code from many system layers
- Provides a flexible framework for adjusting fidelity
  - Based on TinyOS programming model
- Can scale to large numbers

# Outline

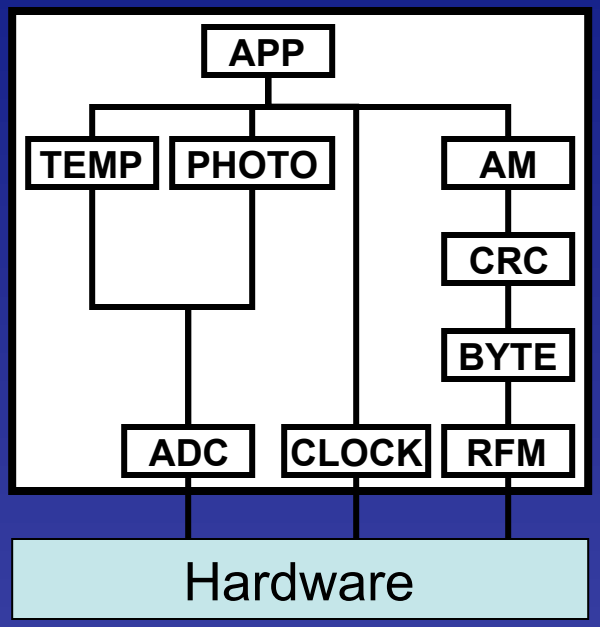
- Introduction
- TOSSIM architecture
- Evaluation
- Conclusion

# Outline

- Introduction
- TOSSIM architecture
- Evaluation
- Conclusion

# Standard TinyOS Program
## (what we start with)

**Component Graph**

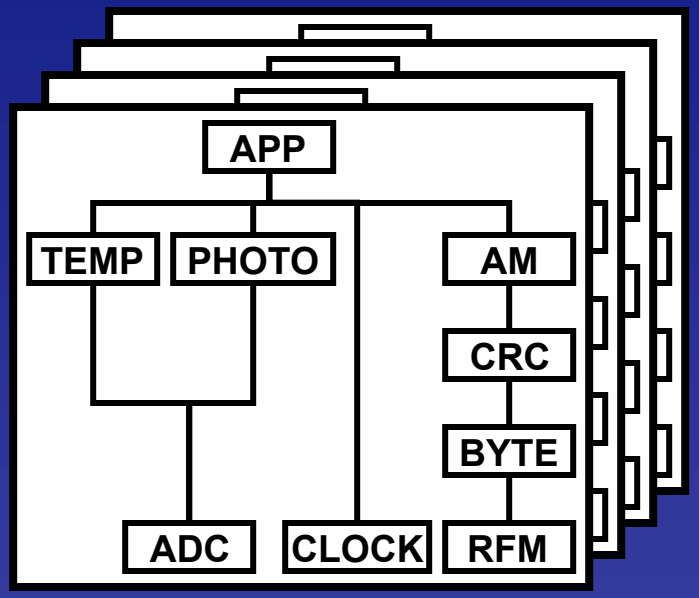


# TOSSIM Adds Five Things

- Alternative compilation target
- Hardware component re-implementations
- Discrete event queue
- Radio and sensor models
- Communication services

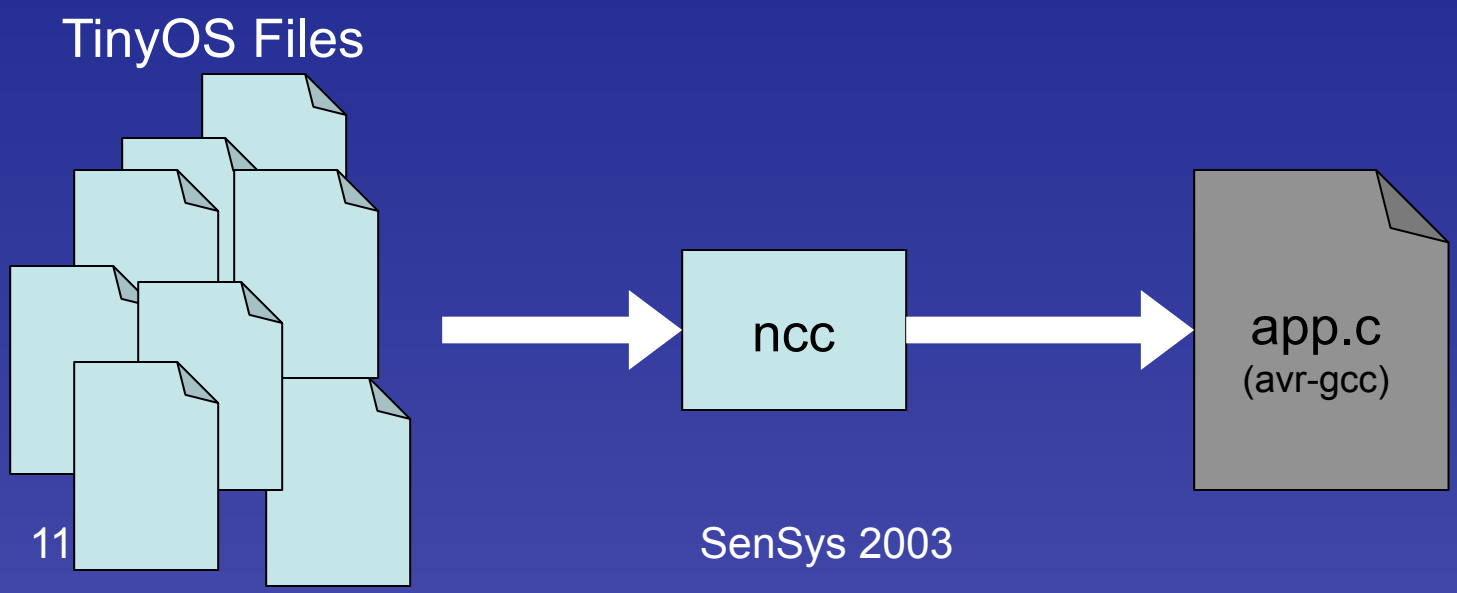
# Multiple Component Graphs

**Component Graphs**



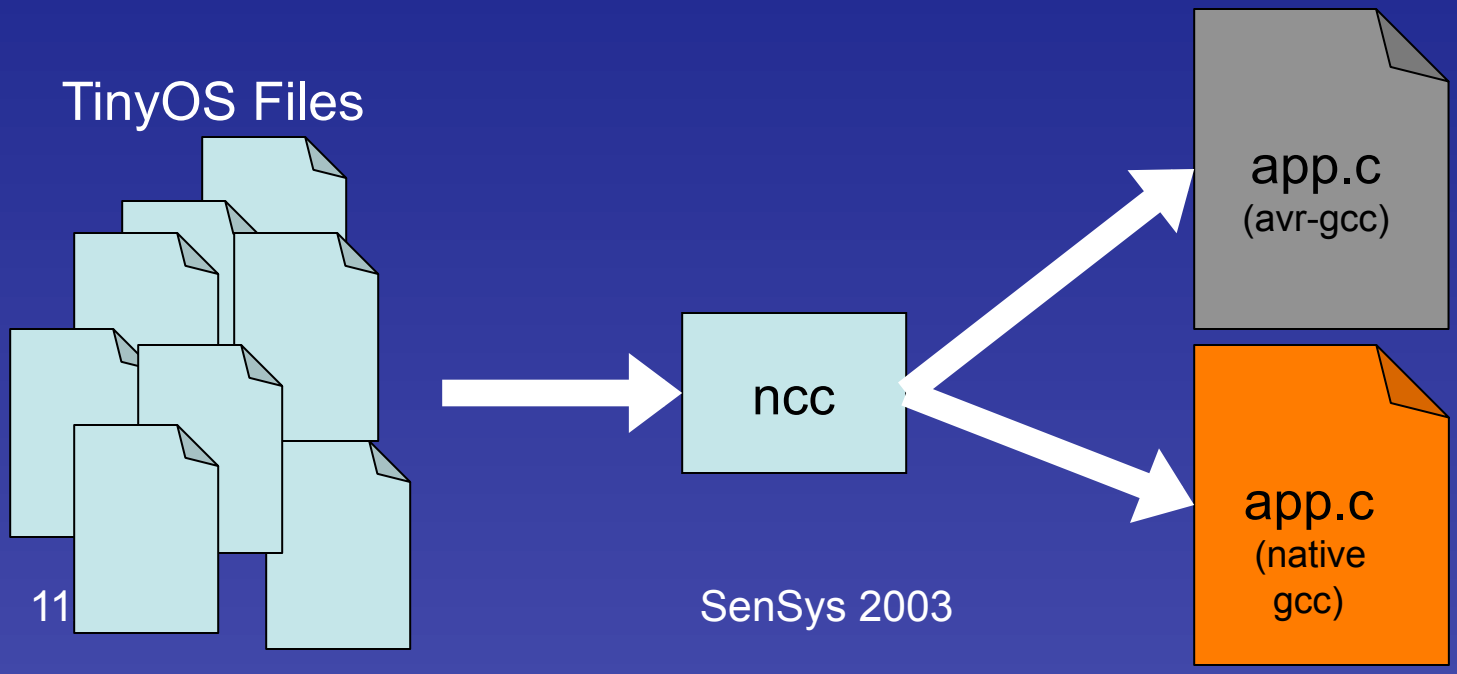
# Compiler Support

- **"Simulator" target added to ncc compiler**
- **Rewrites all state declarations and accesses**
  - **Arrays of variables**
  - **Index into array based on current node**

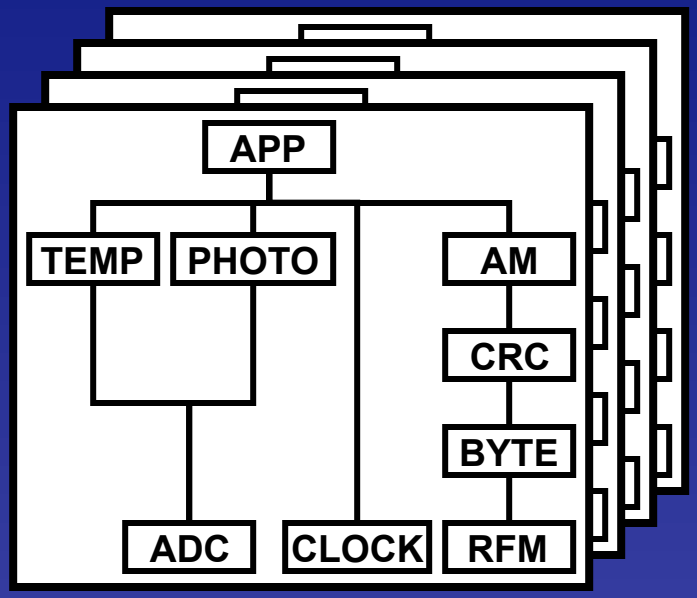


# Compiler Support

- "Simulator" target added to ncc compiler
- Rewrites all state declarations and accesses
  - Arrays of variables
  - Index into array based on current node

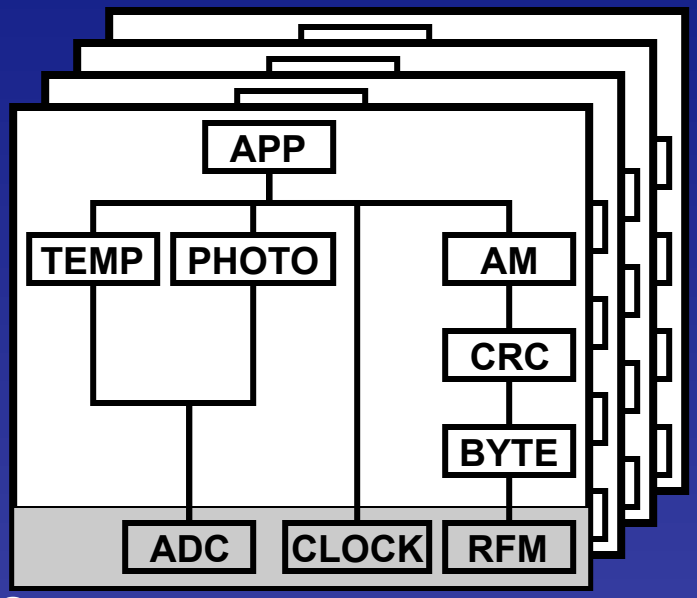


## Component Graphs



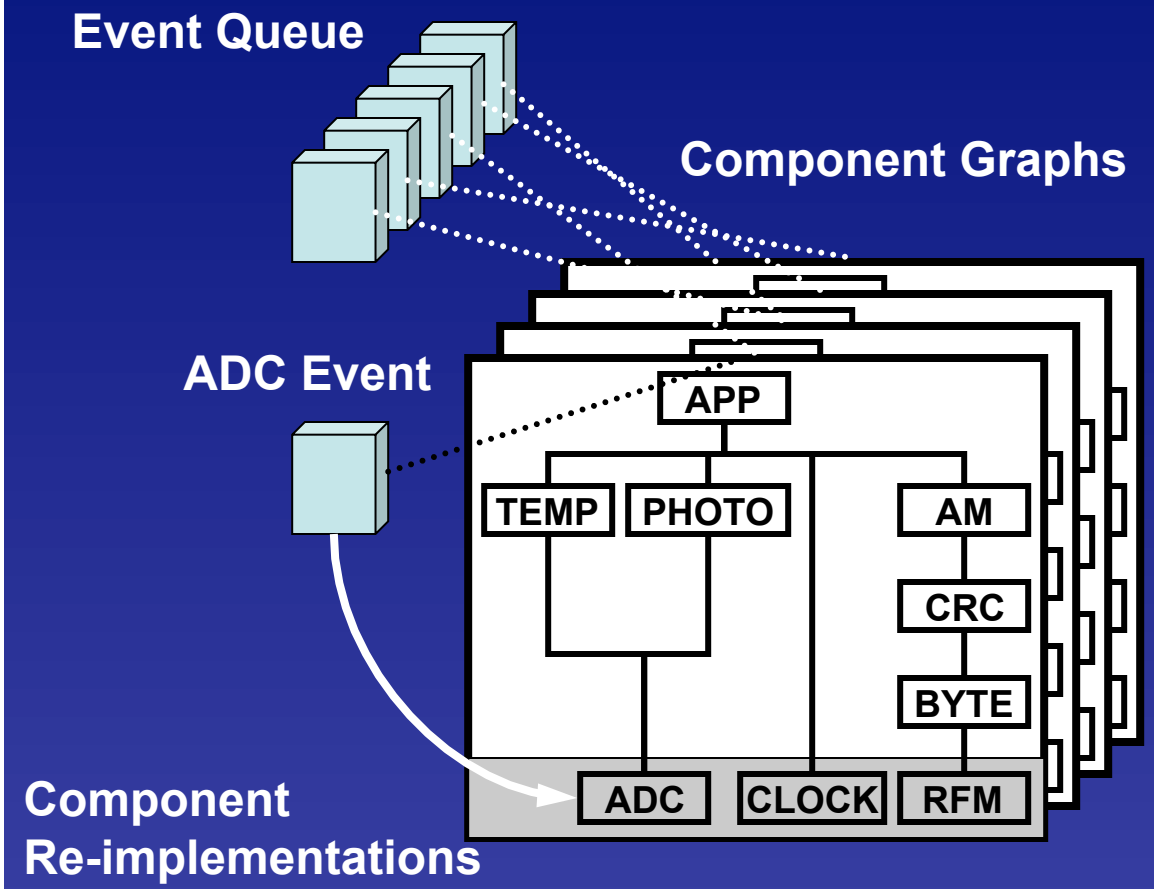
# Component Re-implementations

**Component Graphs**



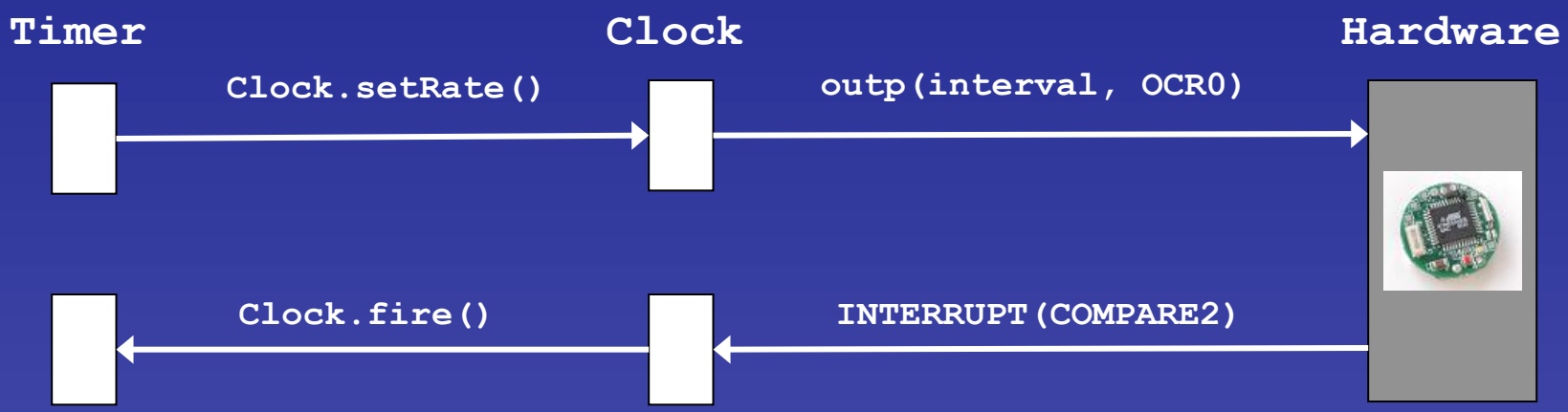

**Component Re-implementations**

# Event Queue


Event Queue
Component Graphs
ADC Event
APP
TEMP
PHOTO
AM
CRC
BYTE
ADC
CLOCK
RFM
Component
Re-implementations

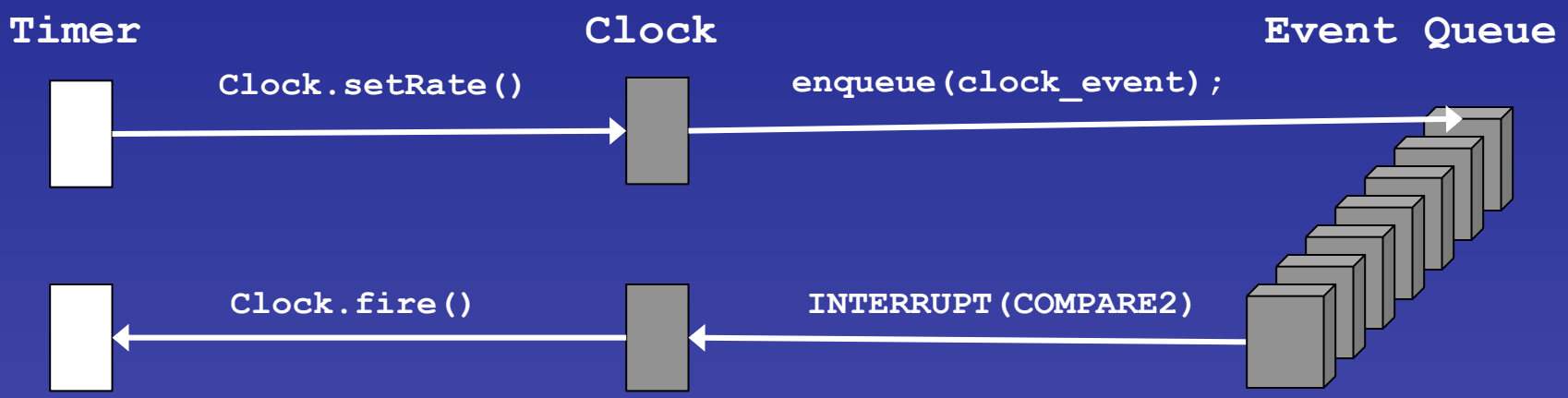
# Mapping TinyOS into TOSSIM

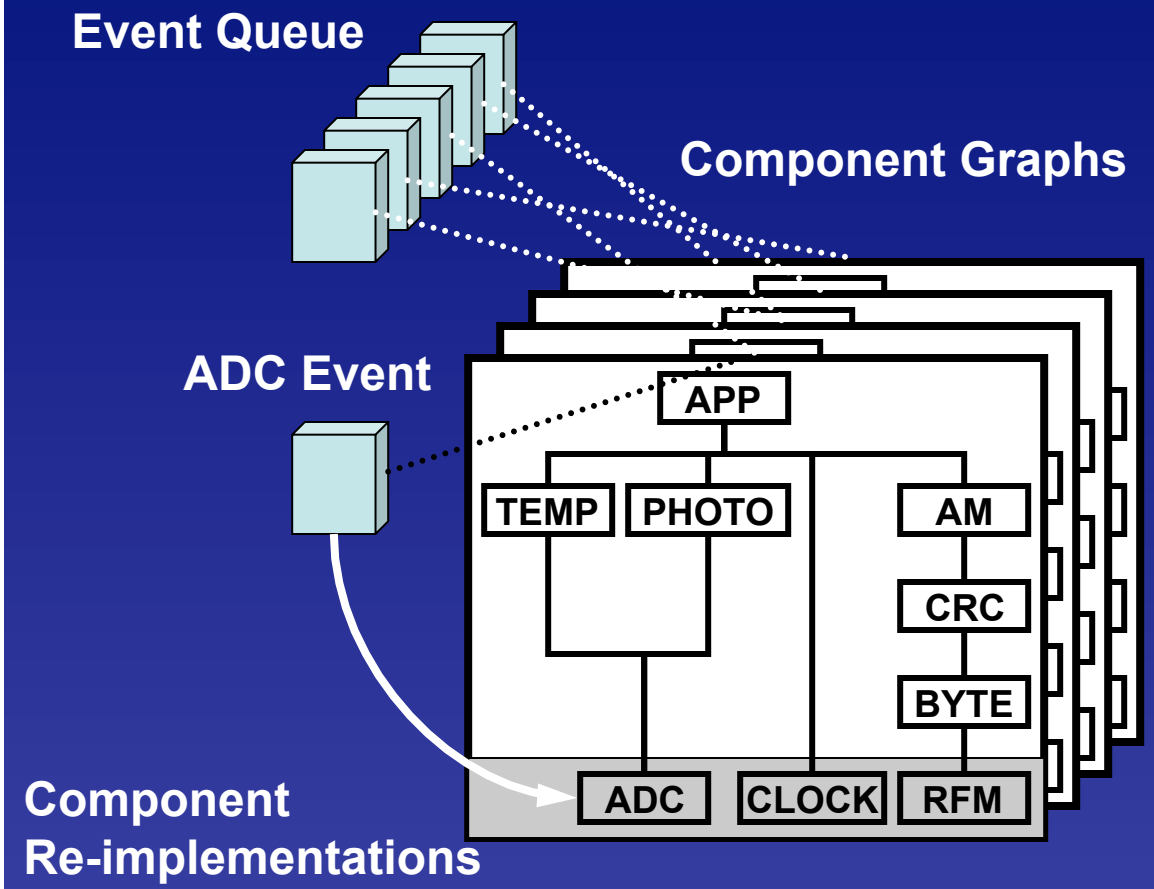
- TOSSIM events trigger TinyOS components
  - Each event is for a specific mote and has a time
- TinyOS calls can enqueue TOSSIM events
- Hardware-level components re-implemented
  - This boundary can change



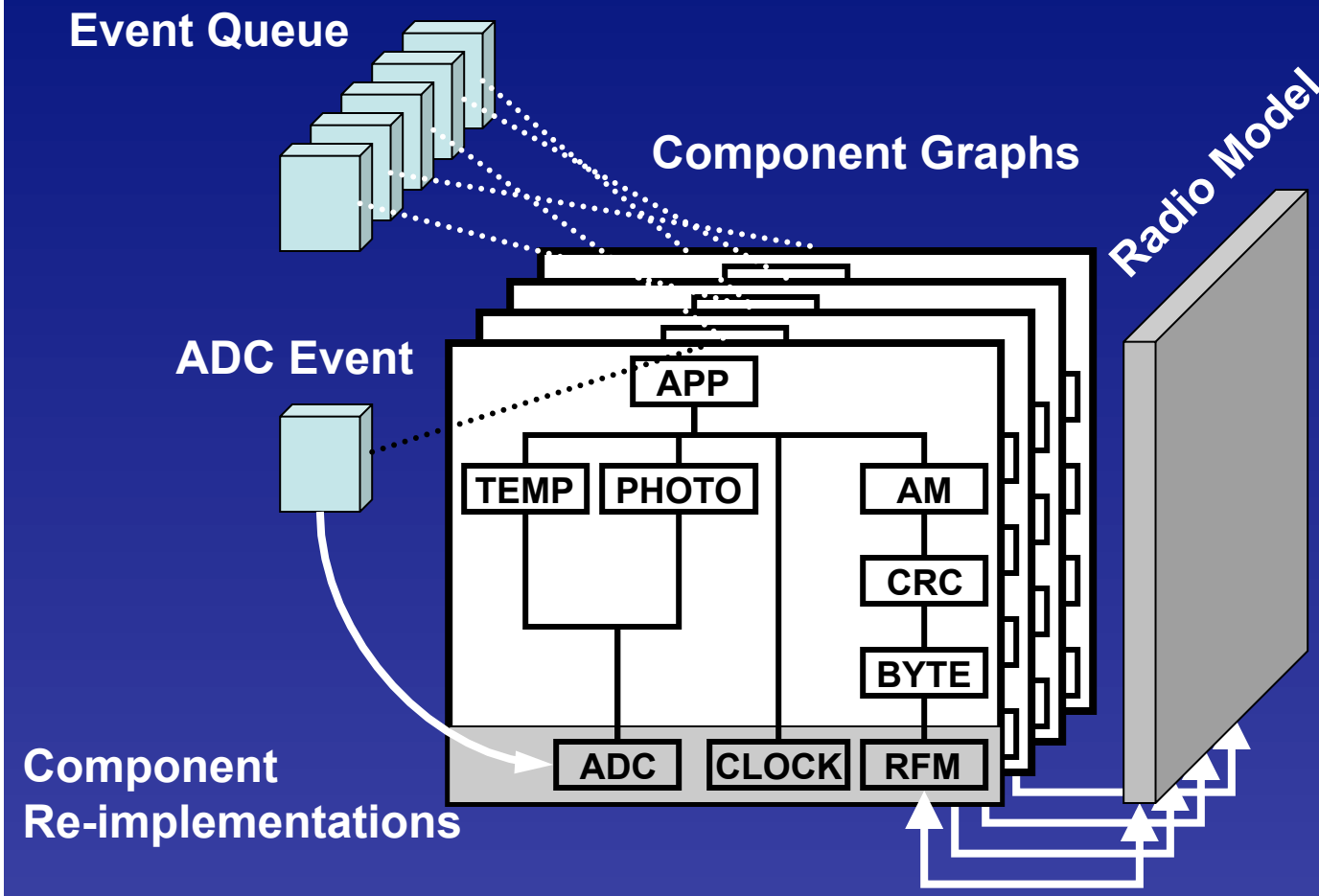
# Mapping TinyOS into TOSSIM

- TOSSIM events trigger TinyOS components
  - Each event is for a specific mote and has a time
- TinyOS calls can enqueue TOSSIM events
- Hardware-level components re-implemented
  - This boundary can change

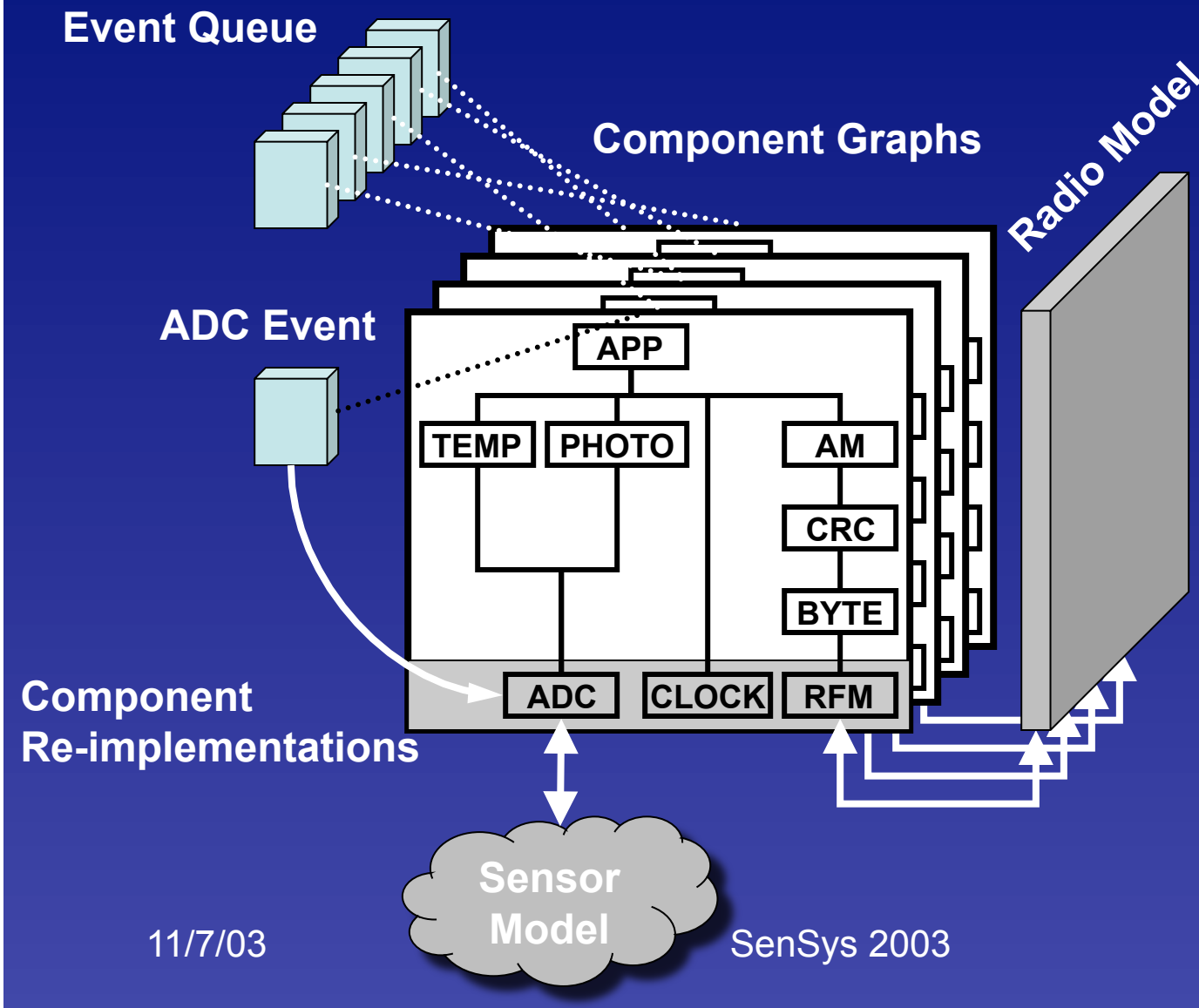



Event Queue
Component Graphs
ADC Event
APP
TEMP
PHOTO
AM
CRC
BYTE
ADC
CLOCK
RFM
Component
Re-implementations

# Radio Model

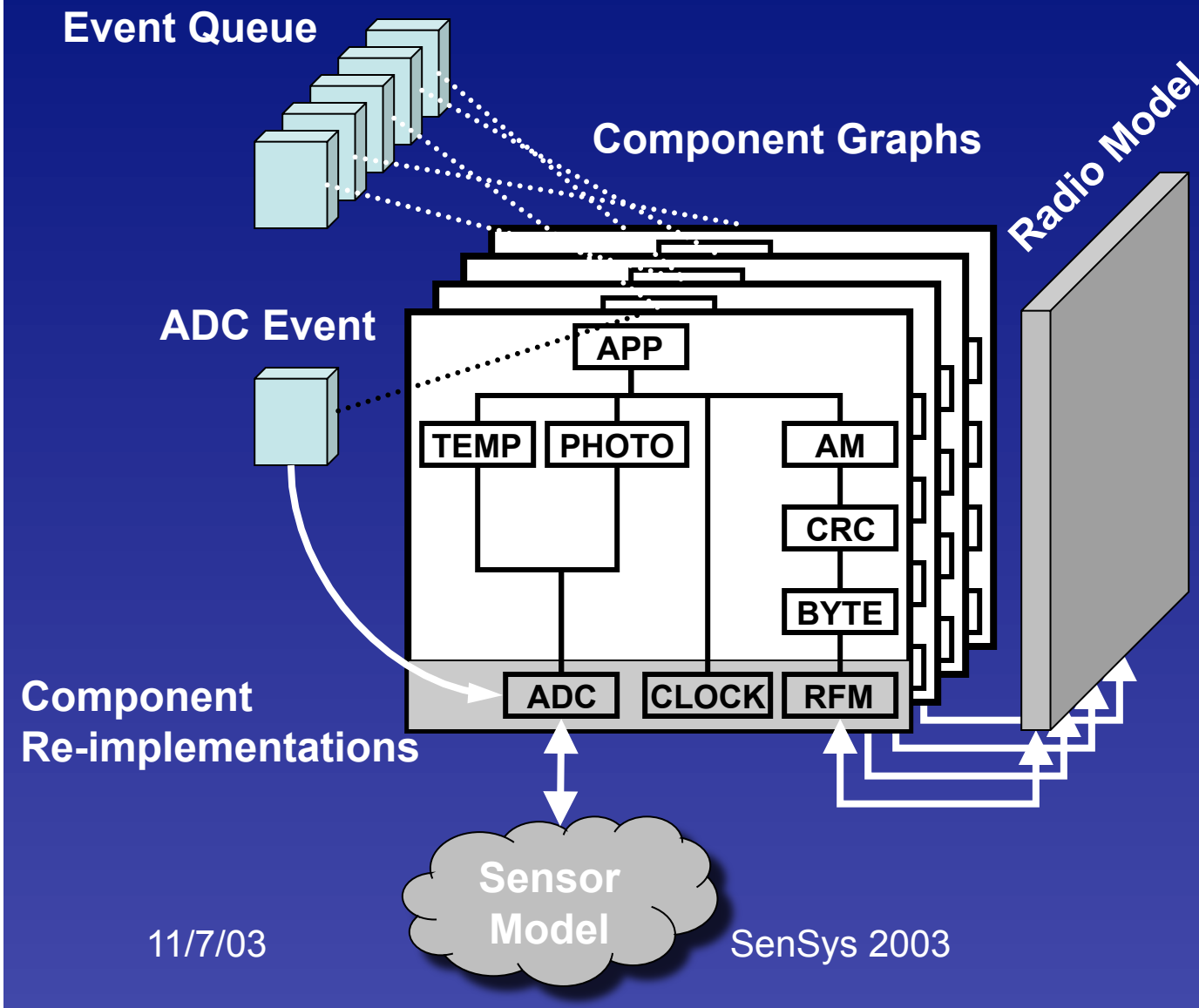

Event Queue
Component Graphs
Radio Model
ADC Event
APP
TEMP
PHOTO
AM
CRC
BYTE
ADC
CLOCK
RFM
Component
Re-implementations

# Sensor Model

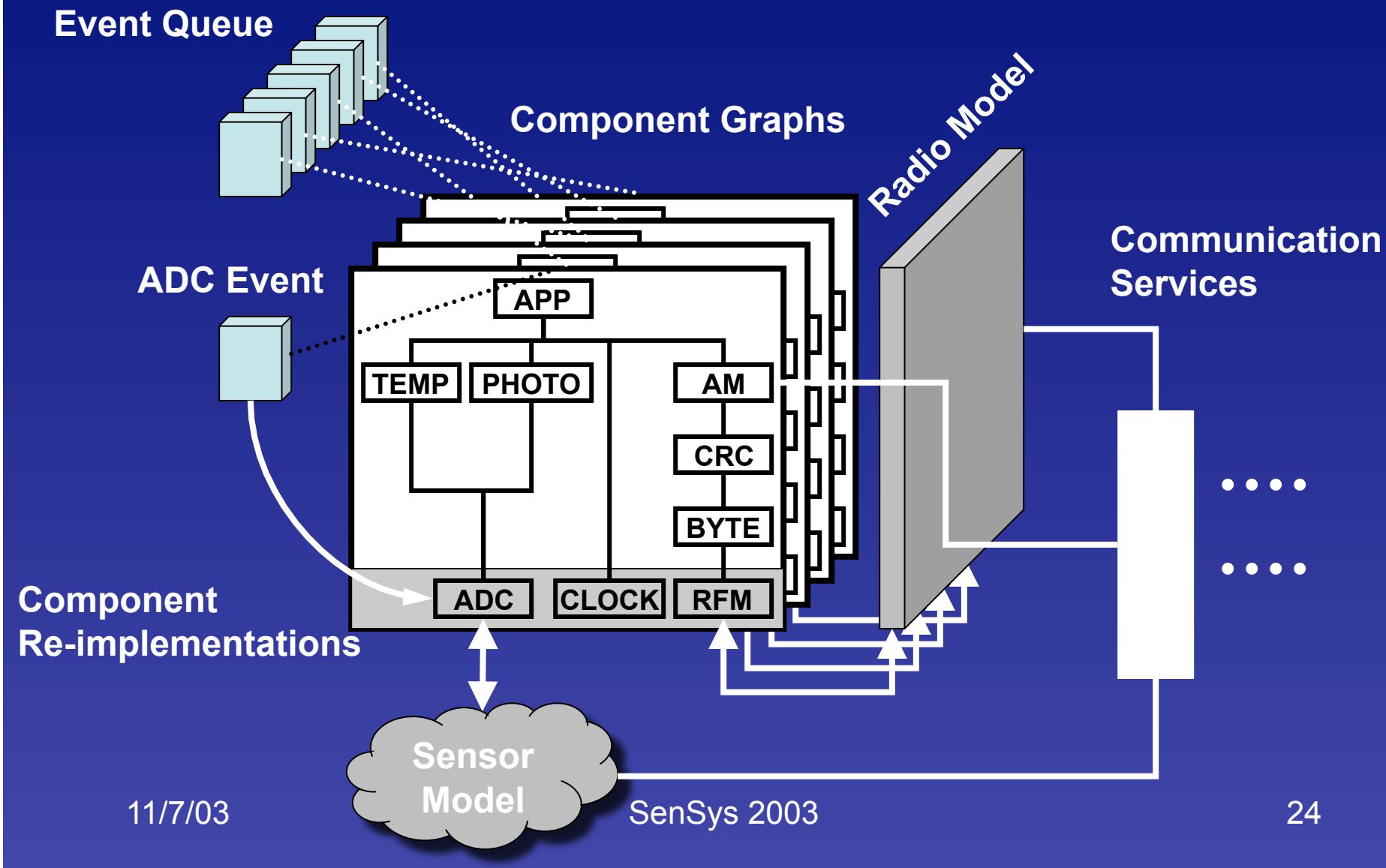


# Radio Model

- Inter-node interaction
- Transmit and receive events
  - Who can hear whom graph
  - Receive radio state
  - Resolution function
  - Individual radio clocks (based on TinyOS code)
- Standard model: directed graph of bit error rates
  - Boil complex models to a connectivity graph
- Standard model: signal is the OR of transmissions
  - All transmissions have uniform strength
  - Hidden terminal problem


Event Queue
Component Graphs
Radio Model
ADC Event
APP
TEMP
PHOTO
AM
CRC
BYTE
ADC
CLOCK
RFM
Component
Re-implementations
Sensor
Model

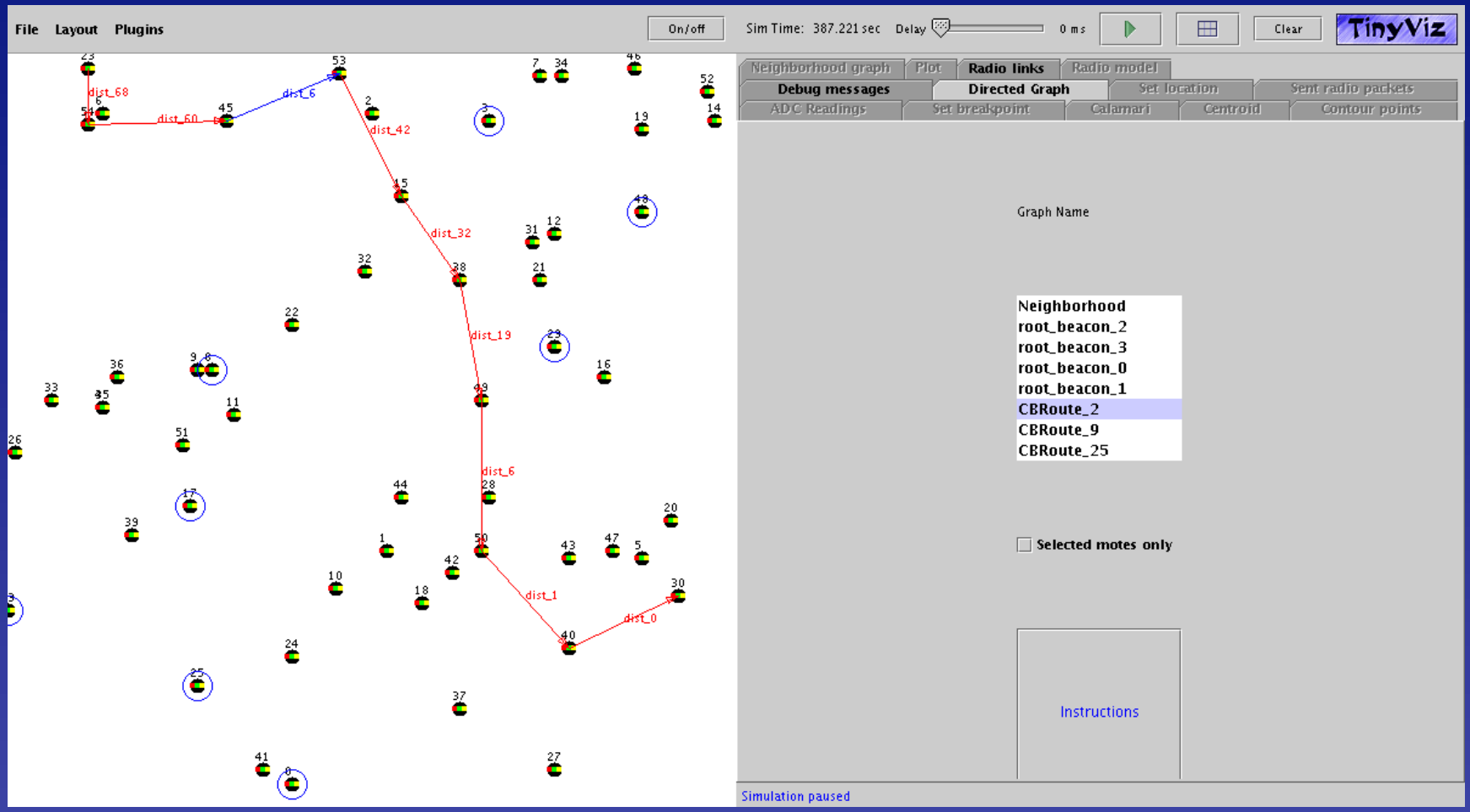
# Actuation and Monitoring



# Actuation and Monitoring

- Programs can actuate and monitor running simulations
  - Inject events, log messages
  - Change simulation state (loss rates, sensors)
- Keeps models external to TOSSIM
  - API to integrate models through sockets
  - TinyViz application

# TinyViz


File
Layout
Plugins
On/off
Sim Time: 387.221 sec
Delay
0 ms
Clear
TinyViz
Neighborhood graph
Plot
Radio links
Radio model
Debug messages
Directed Graph
Set location
Sent radio packets
ADC Readings
Set breakpoint
Calamari
Centroid
Contour points
Graph Name
Neighborhood
root_beacon_2
root_beacon_3
root_beacon_0
root_beacon_1
CBRoute_2
CBRoute_9
CBRoute_25
Selected motes only
Instructions
Simulation paused
dist_68
dist_60
dist_6
dist_42
dist_32
dist_19
dist_6
dist_1
dist_0

# Outline

- Introduction
- TOSSIM architecture
- Evaluation
- Conclusion

# What Do We Need to Capture?
(revisited)

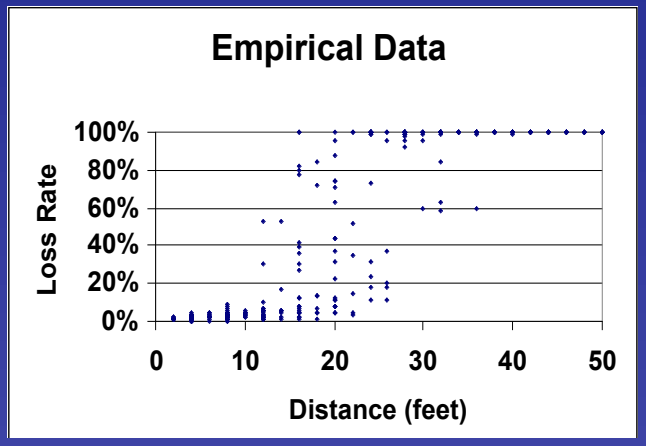
- Bridging
  - Implementations, not just algorithms
- Completeness
  - Spectrum of low-level protocols to applications
  - Applications adapt to their environment
  - Capture the cross-layer interactions
- Fidelity
  - Capture these interactions at a very fine grain
- Scalability
  - Examine behavior in dense or large networks

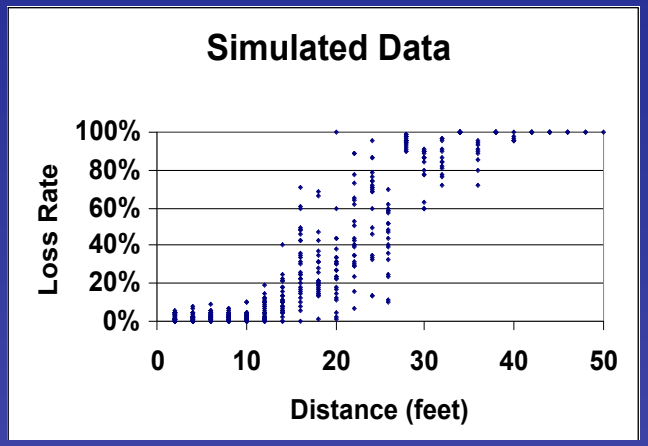
# Evaluation

- **Bridging**
  - **Compiles directly from TinyOS code**
- **Fidelity**
  - **Mote networking**
- **Completeness**
  - **Complete applications**
  - **Unanticipated behavior**
- **Scalability**
  - **Tension between fidelity and scalability**

# Fidelity: Mote Networking

- Generated a loss model from empirical data
  - Map empirical data to TOSSIM loss rates
- Start symbol failure, CRC checks, acknowledgement false negatives/positives
- Ran empirical experiment in TOSSIM



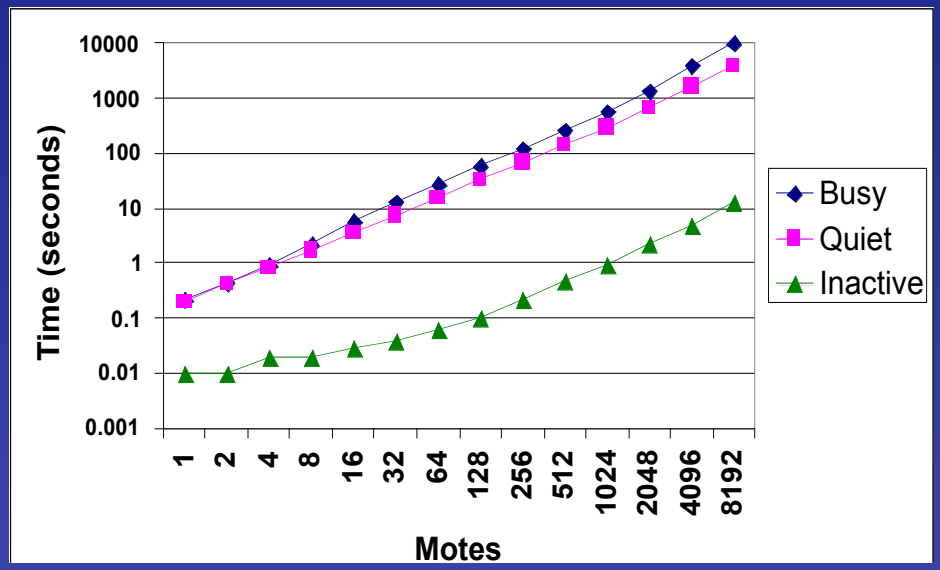



# Completeness

- Network interactions
  - Hidden terminal problem, link asymmetry
  - Discovered bugs in the Surge application
- System-level race conditions
  - Network stack for mica: ChannelMonM start symbol detection and packet transmission
- Testing at low levels
  - State machine rene stack implementation
- Application-level logic driving simulation
  - Maté, TinyDB

# Scalability

- 10 virtual seconds
- Scales to thousands of nodes
- ~30 network active nodes in real time
  - 40,000 radio events per second for each mote



# Scalability Continued

- 30 nodes * 40,000 events = 1.2 million TOSSIM events/second
- Bit-level simulation is the bottleneck
  - But we can move the re-implementation line…

# Outline

- Introduction
- TOSSIM architecture
- Evaluation
- Conclusion

# Some Cautionary Notes

- Non-preemptive execution
  - Handlers execute instantaneously
- Signal strength
  - Results and assumptions
- Independent bit errors
- Perfect channel sense
- TOSSIM has not been validated
  - A basis for comparison, but not *the* basis

# Extensions
## (help appreciated!)

- Changing the re-implementation boundary
  - Packet level simulation
  - Talking to real motes
- Heterogeneous networks
  - Additional compiler support
  - Compelling applications?
- Power modeling
  - Source annotation for CPU timing
  - Component power state transitions

# Concluding Thoughts

- Event-driven OS maps to event-driven simulation
- TOSSIM as a TinyOS *development environment*
  - Validate with deployment
- Simulate at the lowest level
  - More than what you think you need
  - Capture the *unanticipated* problems
- Capture the mess, but not reality
  - Model the observed behavior, not the phenomena that cause that behavior
  - Do we understand these systems well enough yet?

# Questions

# Backup Slides

# Why Not ns-2?

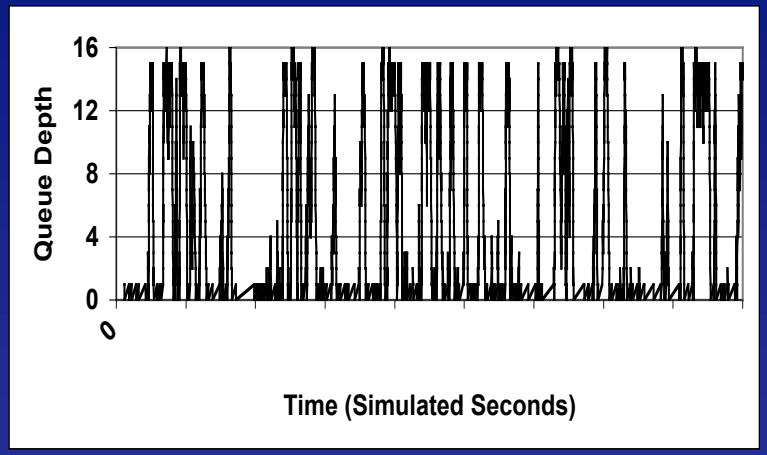
- Not designed for application interaction
- Algorithms, not implementations
- Need a TinyOS simulator, not just the network

# Surge

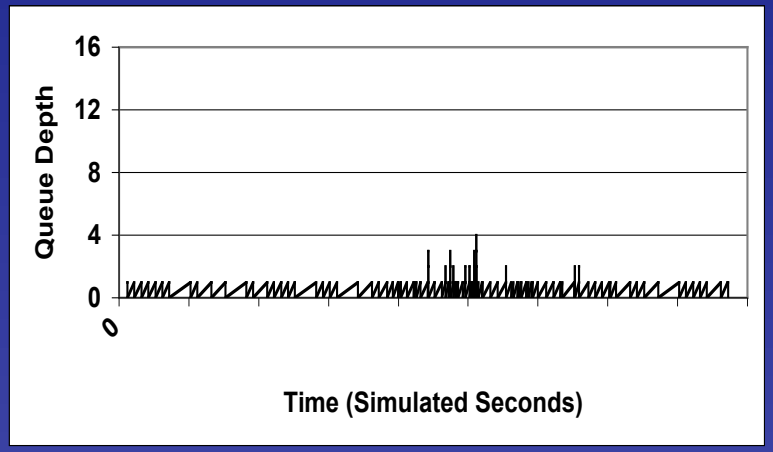
- Empirically, Surge protocol showed poor performance
  - Users added TinyDB attribute to monitor protocol
  - Everything looked fine
- Tested Surge application in TOSSIM
  - Observed traffic surges
  - Send queue overflow
- TOSSIM revealed the problem
  - Cycles + acks + retransmission policy
  - Lose TinyDB query results indicating the problem

# Surge, Continued

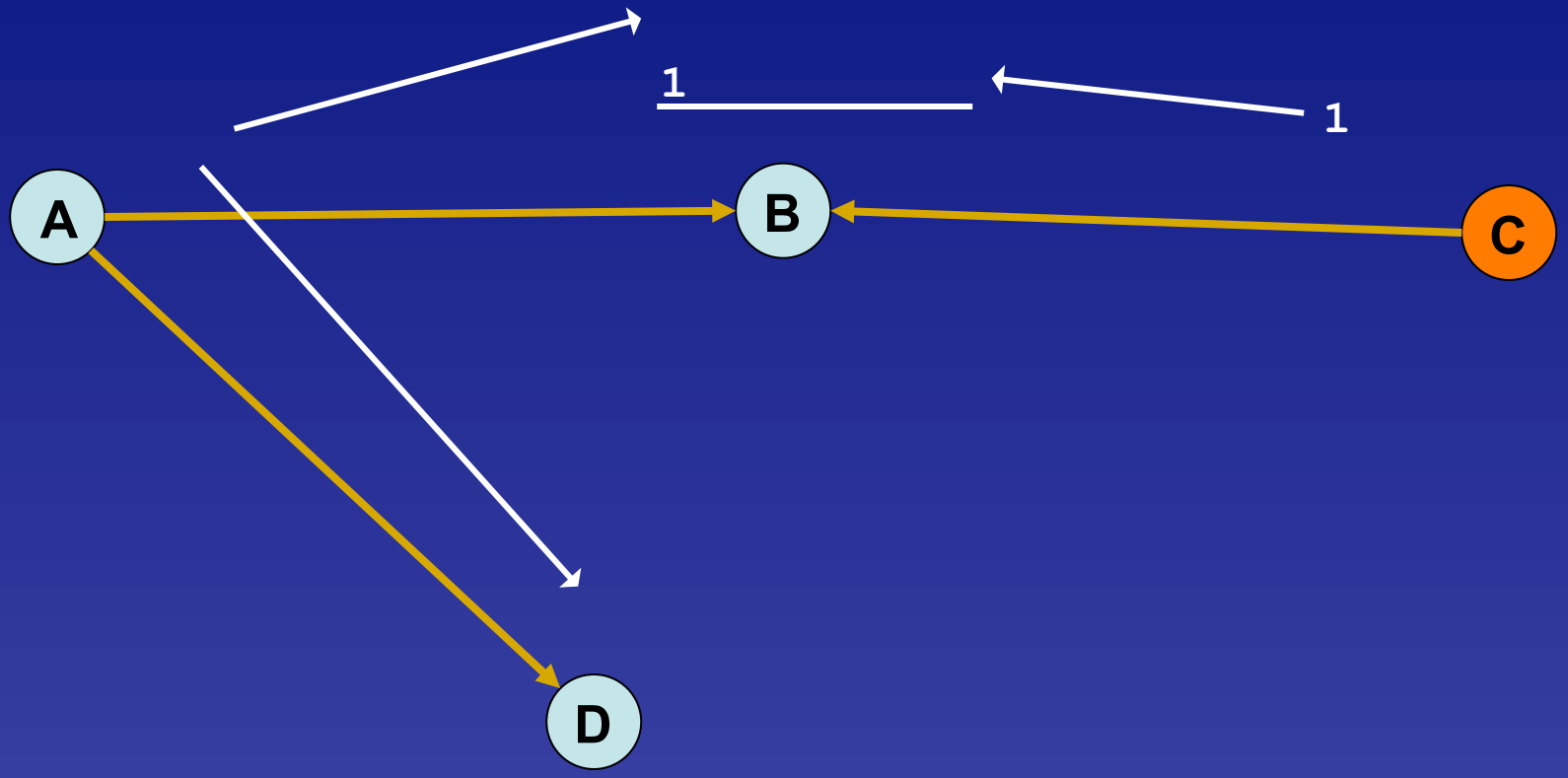
Observed Behavior



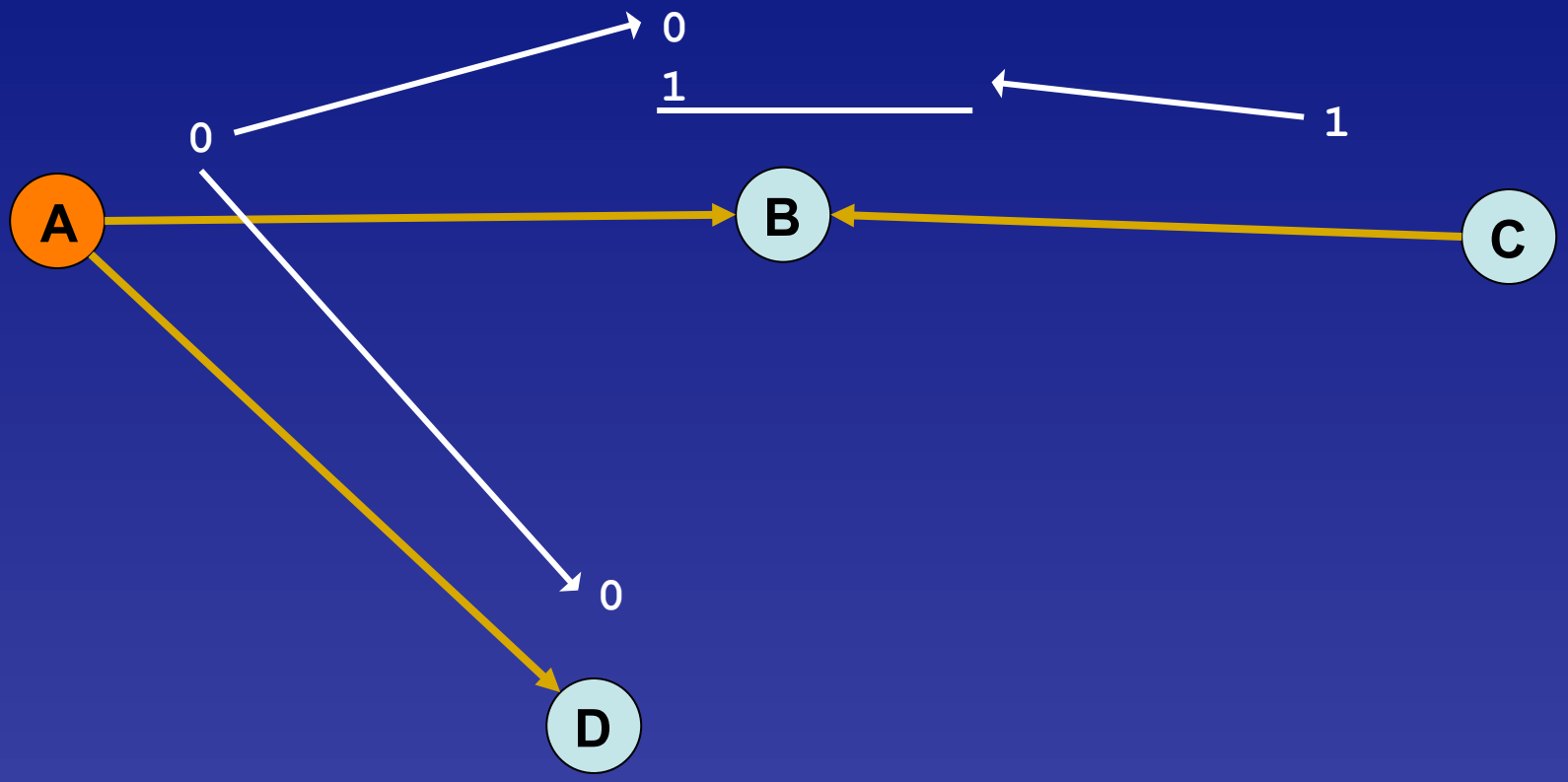

Fixed Algorithm



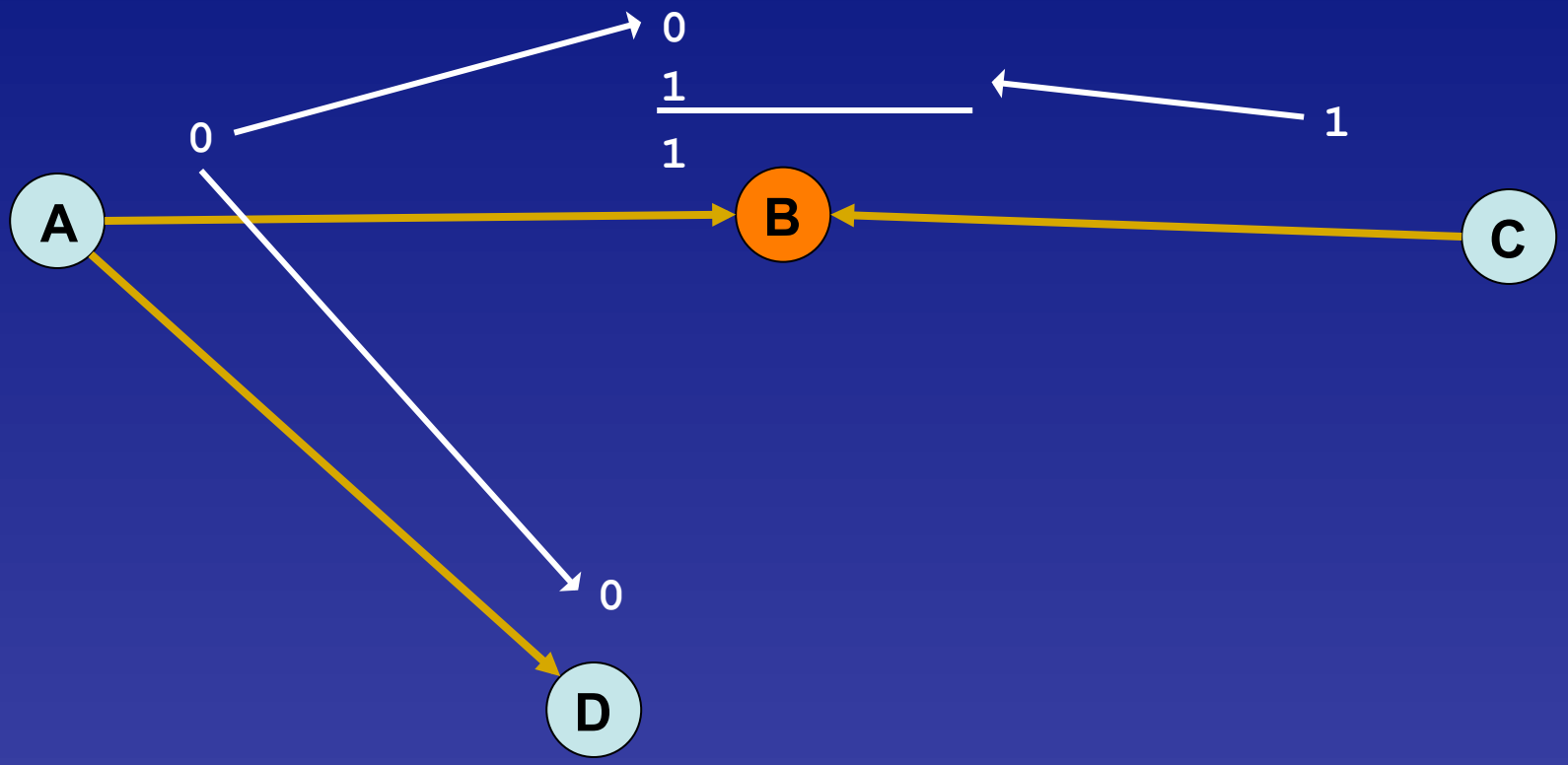
# Radio Example



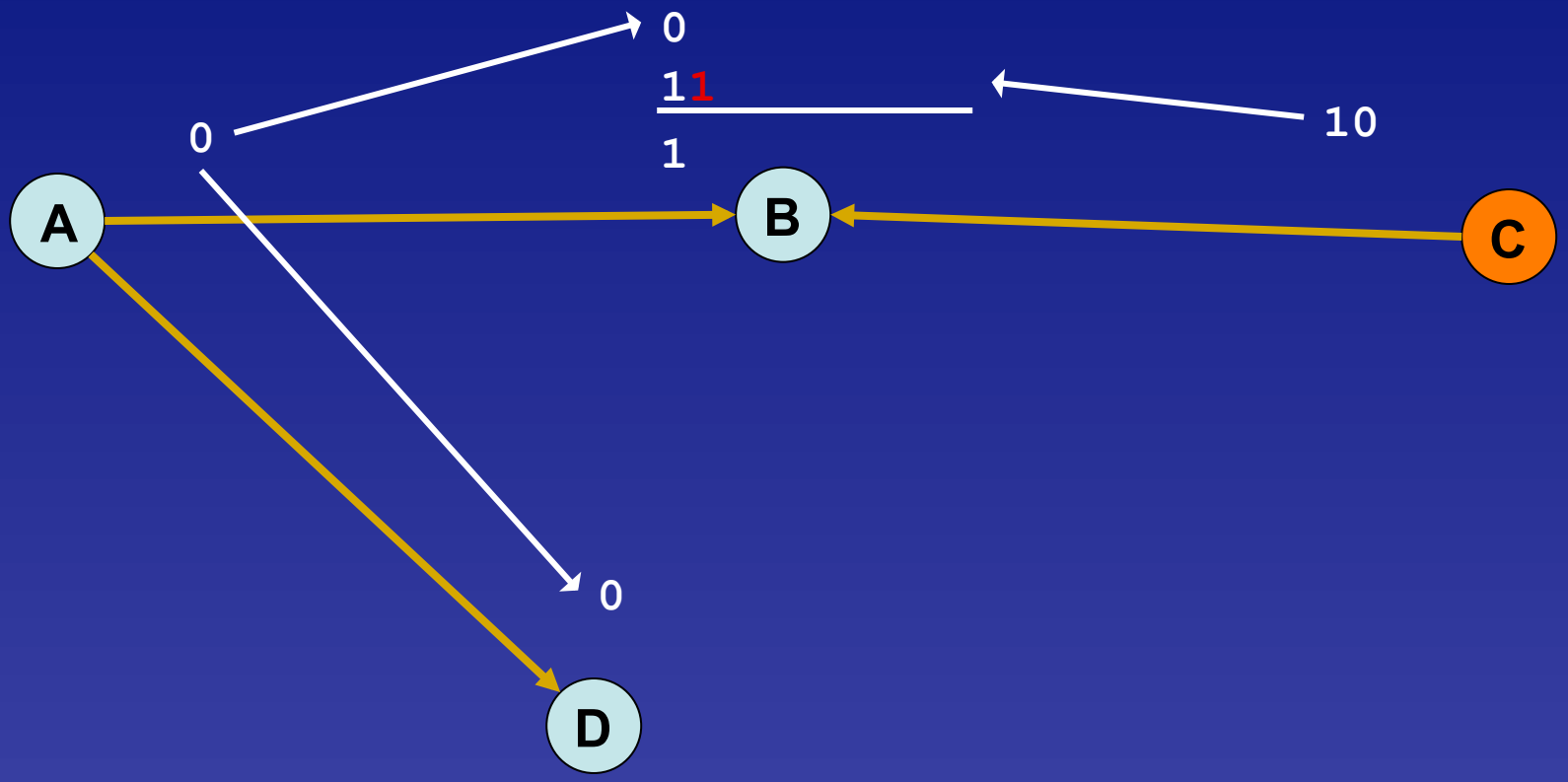
# Radio Example



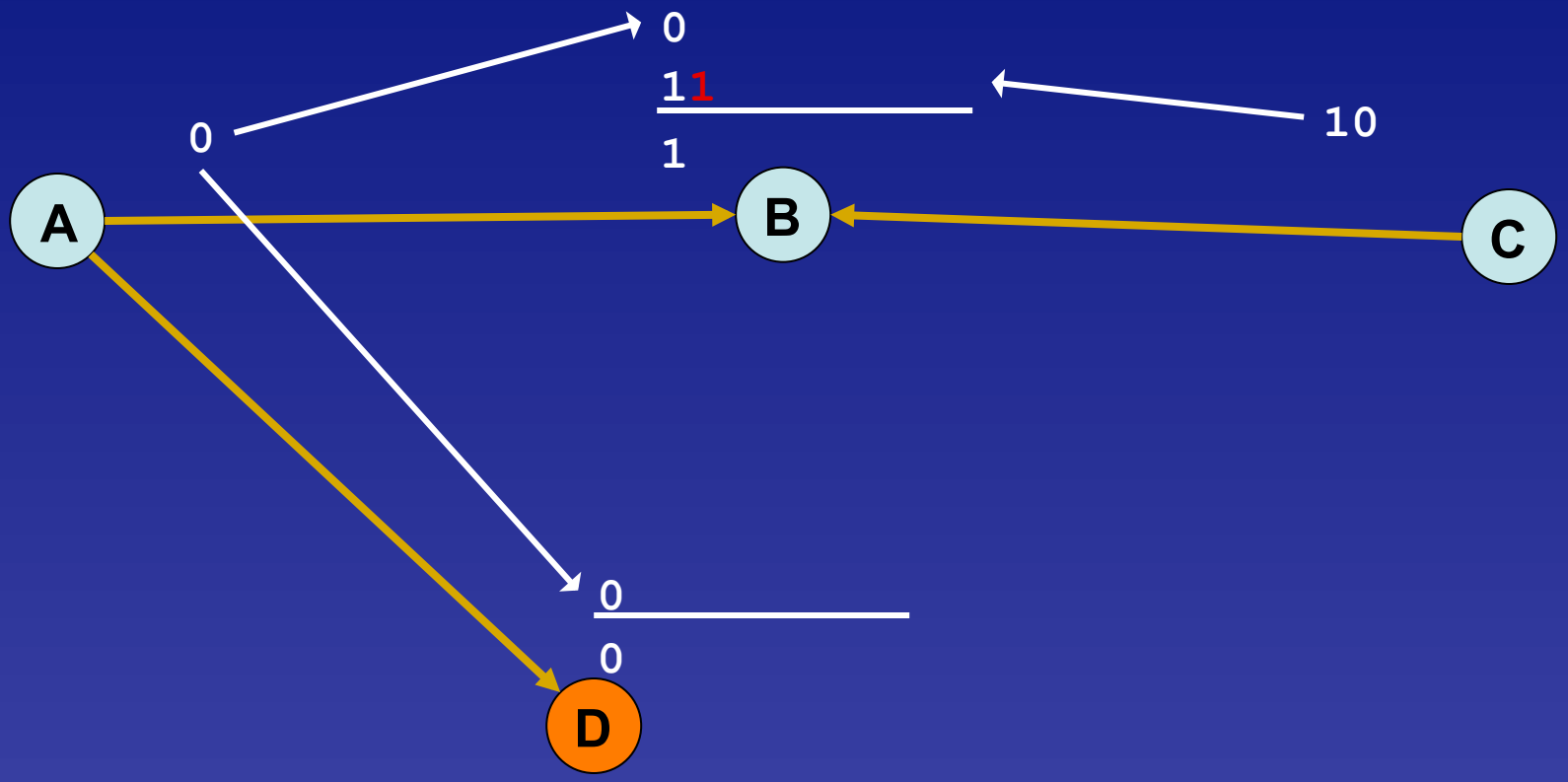
# Radio Example



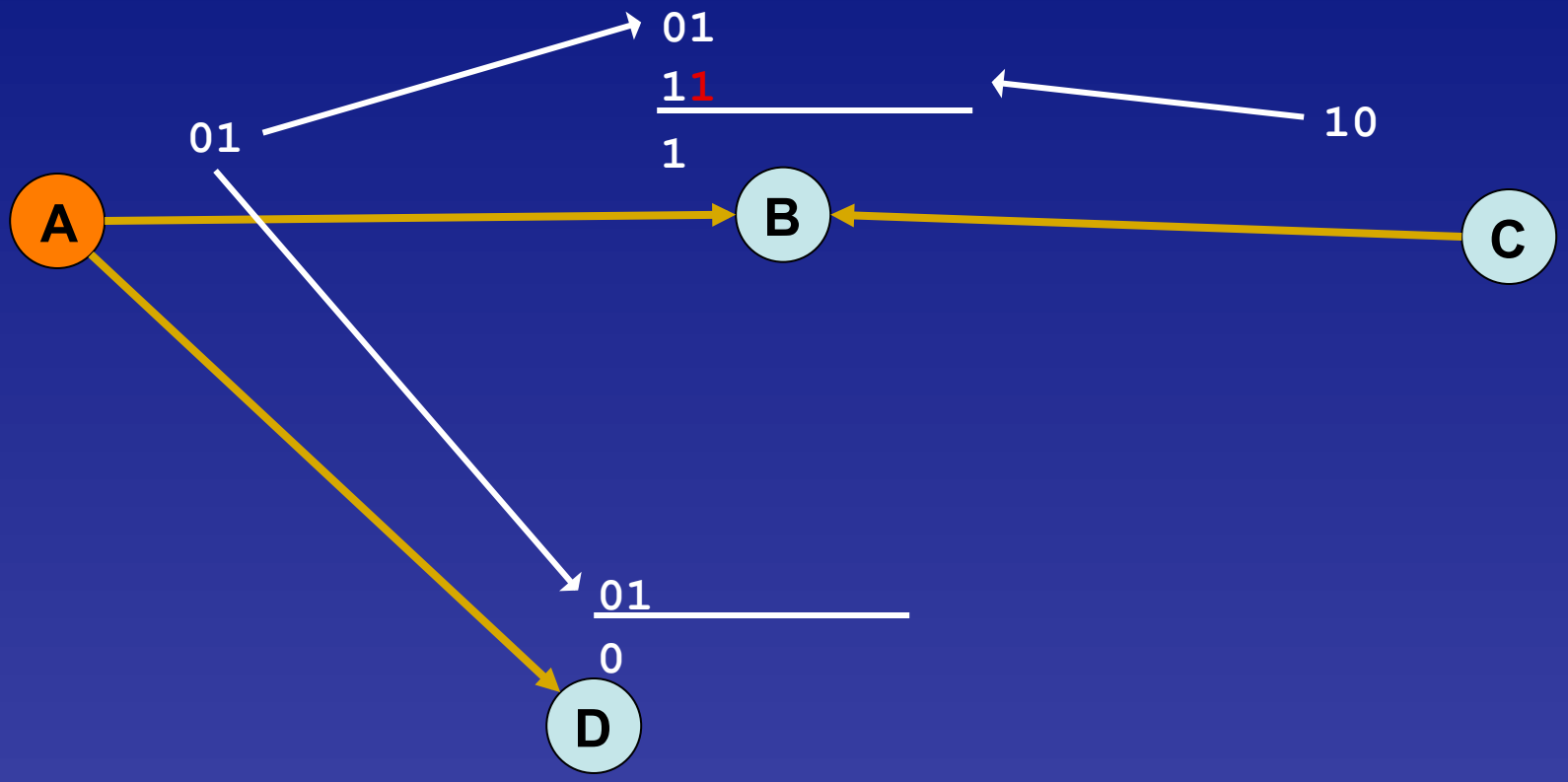
# Radio Example



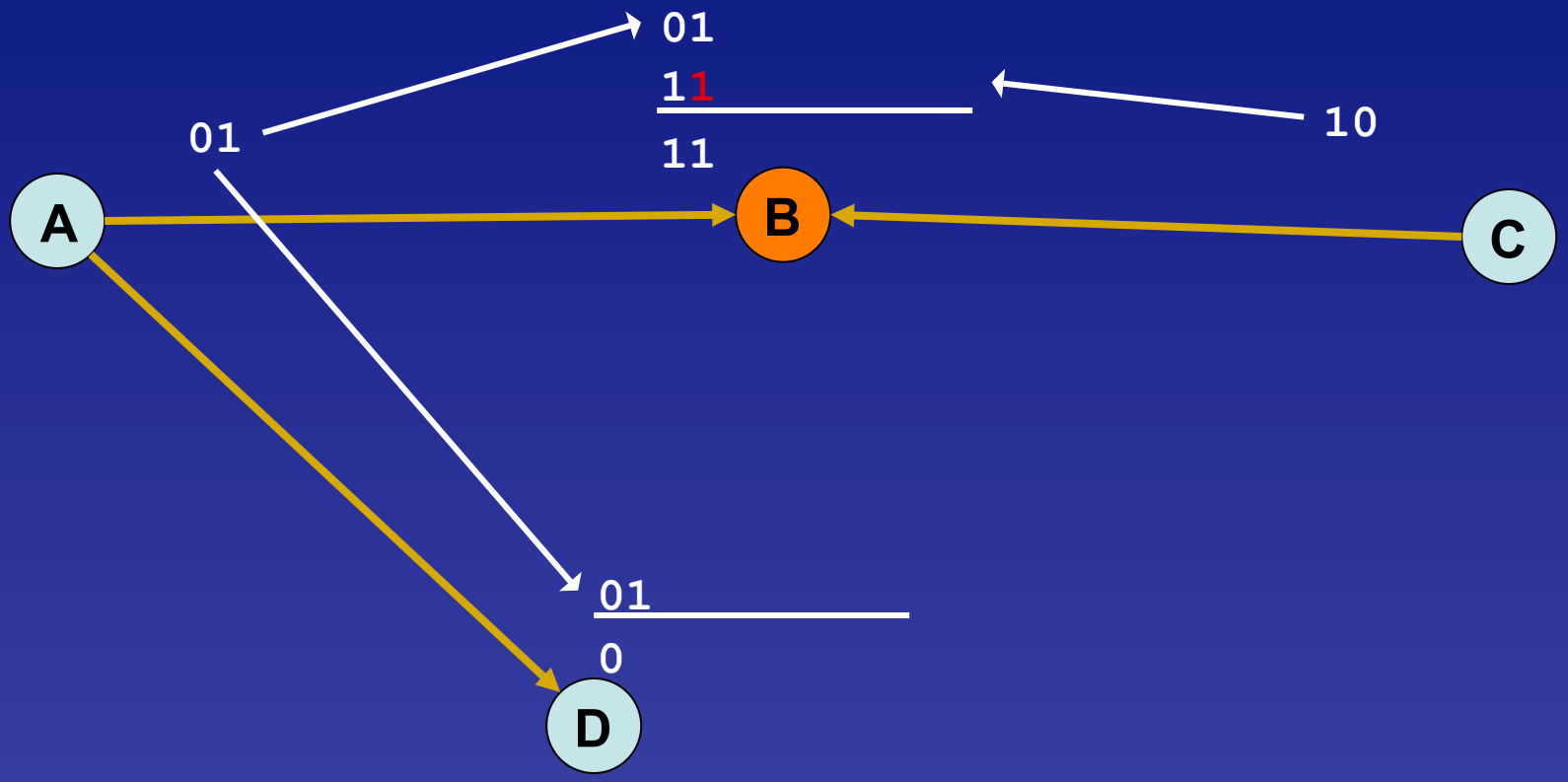
# Radio Example



# Radio Example



# Radio Example



# Radio Example