Race Detection for Android

Midterm Report

Tian Lan

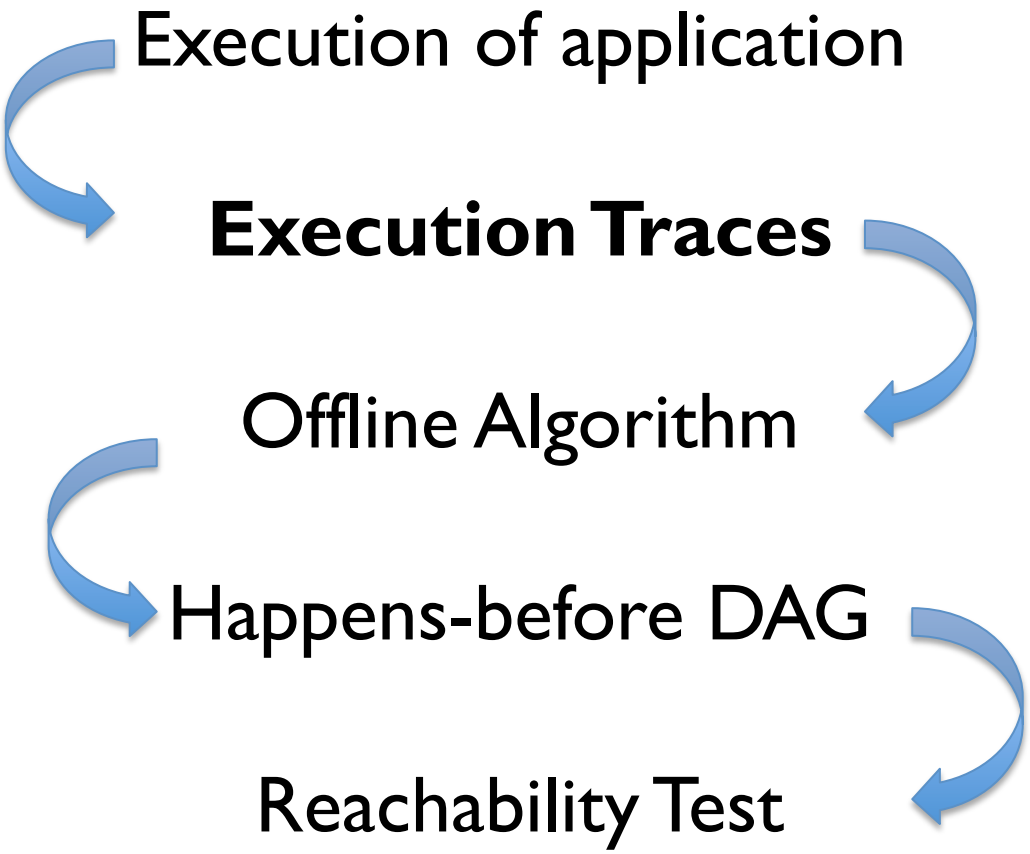
Android Event-Driven Model

- Asynchronous event
  - generated by:
    - Externally: Sensor input, network, etc
    - Internally: Thread or event
  - handled by: *event handler*
  - organized by: *looper* thread
  - reordered non-deterministically
  - once started, runs to completion
  - make synchronous and asynchronous calls

Android Concurrency Model

- Looper thread:
  - associated with one event queue:
    - Holds all the events in the looper thread
    - FIFO (affected by event time constraints)
  - continuously check its event queue:
    - Select and process one at a time
    - Events in the same looper thread are atomic w.r.t each other

Approach

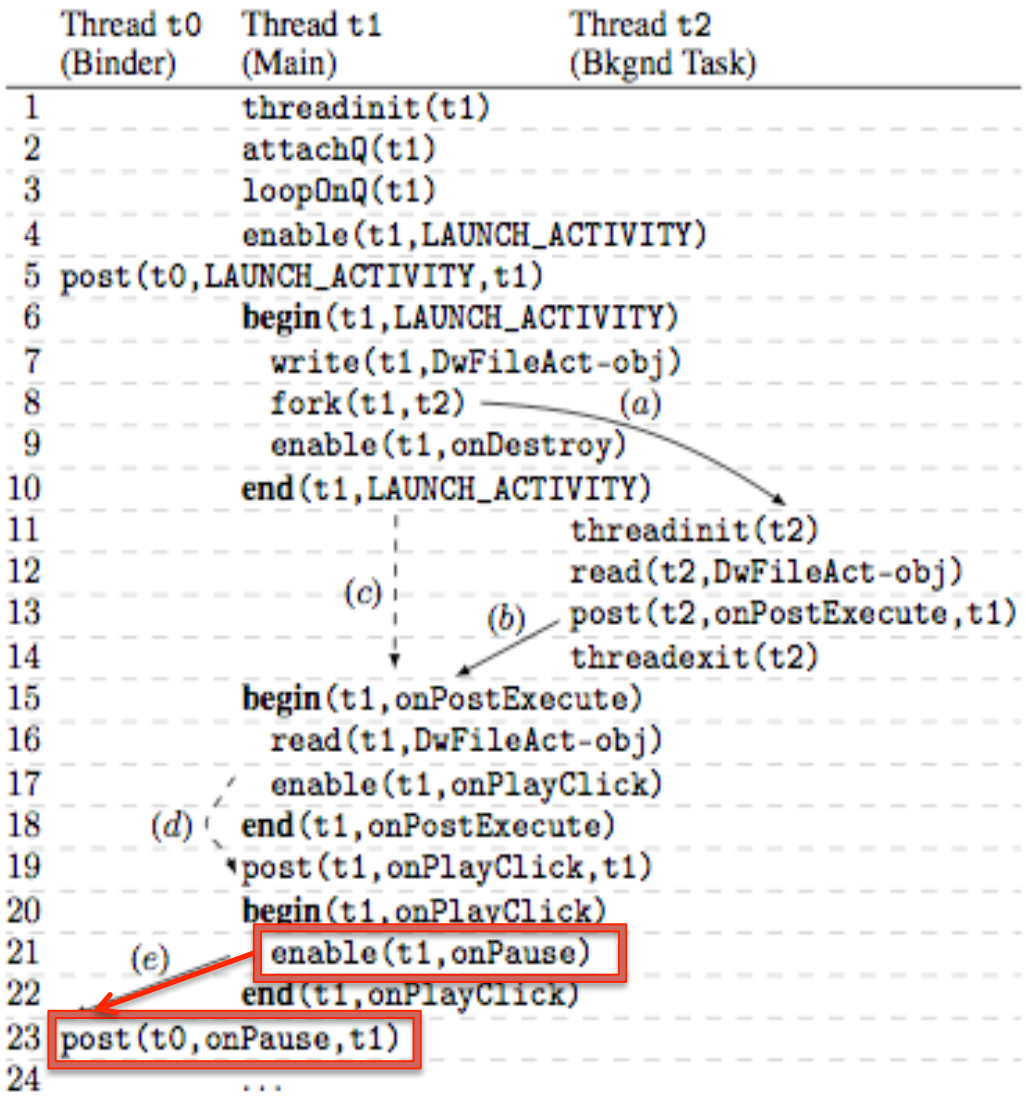

Execution of application
Execution Traces
Offline Algorithm
Happens-before DAG
Reachability Test

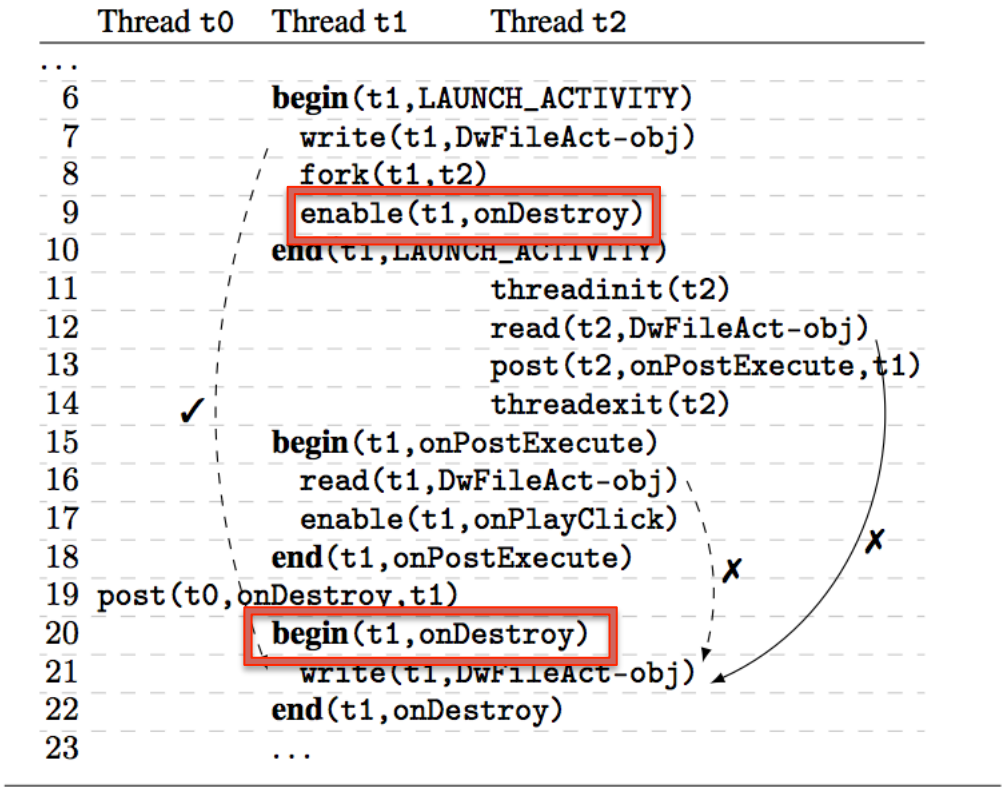
Execution Trace

Race Detection for Android Applications, *PLDI 14*

- threadinit(t), threadexit(t)
- fork(t, t0), join(t, t0)
- acquire(t, l), release(t, l)
- read(t, m), write(t, m)

- *attachQ(t)*: attach a task queue to thread t
- *loopOnQ(t)*: begin executing procedures in t's queue
- *begin(t, p)*, end(t, p): start/end executing the posted task p
- ***enable(t, p)***: enable posting of task p
- ***post(t, p, t0)***: post task p asynchronously to thread t0


Thread t0 (Binder)
Thread t1 (Main)
Thread t2 (Bkgnd Task)
1 threadinit(t1)
2 attachQ(t1)
3 loopOnQ(t1)
4 enable(t1,LAUNCH_ACTIVITY)
5 post(t0,LAUNCH_ACTIVITY,t1)
6 begin(t1,LAUNCH_ACTIVITY)
7 write(t1,DwFileAct-obj)
8 fork(t1,t2)
(a)
9 enable(t1,onDestroy)
10 end(t1,LAUNCH_ACTIVITY)
11 threadinit(t2)
12 read(t2,DwFileAct-obj)
(c)
13 post(t2,onPostExecute,t1)
(b)
14 threadexit(t2)
15 begin(t1,onPostExecute)
16 read(t1,DwFileAct-obj)
17 enable(t1,onPlayClick)
18 (d) end(t1,onPostExecute)
19 post(t1,onPlayClick,t1)
20 begin(t1,onPlayClick)
21 (e) enable(t1,onPause)
22 end(t1,onPlayClick)
23 post(t0,onPause,t1)
24 ...


Thread t0
Thread t1
Thread t2
...
6 begin(t1,LAUNCH_ACTIVITY)
7 write(t1,DwFileAct-obj)
8 fork(t1,t2)
9 enable(t1,onDestroy)
10 end(t1,LAUNCH_ACTIVITY)
11 threadinit(t2)
12 read(t2,DwFileAct-obj)
13 post(t2,onPostExecute,t1)
14 threadexit(t2)
15 begin(t1,onPostExecute)
16 read(t1,DwFileAct-obj)
17 enable(t1,onPlayClick)
18 end(t1,onPostExecute)
19 post(t0,onDestroy,t1)
20 begin(t1,onDestroy)
21 write(t1,DwFileAct-obj)
22 end(t1,onDestroy)
23 ...
✓
✗
✗

Happens-before rule

Race Detection for Event-Driven Mobile Applications, *PLDI 14*

- Thread-local happens-before:
- Inter-thread happens-before:

$$\text{(NO-Q-PO)}\ \frac{\texttt{loopOnQ}(\texttt{t}) \notin \{\alpha_1, \ldots, \alpha_{i-1}\} \quad \mathit{thread}(\alpha_i) = \mathit{thread}(\alpha_j) = \texttt{t}}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(ASYNC-PO)}\ \frac{\texttt{loopOnQ}(\texttt{t}) \in \{\alpha_1, \ldots, \alpha_{i-1}\} \quad \mathit{task}(\alpha_i) = \mathit{task}(\alpha_j) = (\texttt{t}, \_)}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(ENABLE-ST)}\ \frac{\alpha_i = \texttt{enable}(\texttt{t}, \texttt{p}) \qquad \alpha_j = \texttt{post}(\texttt{t}, \texttt{p}, \_)}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(POST-ST)}\ \frac{\alpha_i = \texttt{post}(\texttt{t}, \texttt{p}, \texttt{t}) \qquad \alpha_j = \texttt{begin}(\texttt{t}, \texttt{p})}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(FIFO)}\ \frac{\alpha_i = \texttt{end}(\texttt{t}, \texttt{p1}) \quad \alpha_j = \texttt{begin}(\texttt{t}, \texttt{p2}) \quad \texttt{post}(\_, \texttt{p1}, \texttt{t}) \preceq \texttt{post}(\_, \texttt{p2}, \texttt{t})}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(NOPRE)}\ \frac{\alpha_i = \texttt{end}(\texttt{t}, \texttt{p1}) \qquad \alpha_j = \texttt{begin}(\texttt{t}, \texttt{p2}) \quad \exists \alpha_k.\ \mathit{task}(\alpha_i) = \mathit{task}(\alpha_k) \wedge \alpha_k \preceq \texttt{post}(\_, \texttt{p2}, \texttt{t})}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(TRANS-ST)}\ \frac{\alpha_i \preceq_{st} \alpha_k \qquad \alpha_k \preceq_{st} \alpha_j}{\alpha_i \preceq_{st} \alpha_j}$$

$$\text{(ATTACH-Q-MT)}\ \frac{\alpha_i = \texttt{attachQ}(\texttt{t}) \qquad \alpha_j = \texttt{post}(\texttt{t}', \_, \texttt{t})}{\alpha_i \preceq_{mt} \alpha_j}$$

$$\text{(ENABLE-MT)}\ \frac{\alpha_i = \texttt{enable}(\texttt{t}, \texttt{p}) \qquad \alpha_j = \texttt{post}(\texttt{t}', \texttt{p}, \_)}{\alpha_i \preceq_{mt} \alpha_j}$$

$$\text{(POST-MT)}\ \frac{\alpha_i = \texttt{post}(\texttt{t}', \texttt{p}, \texttt{t}) \qquad \alpha_j = \texttt{begin}(\texttt{t}, \texttt{p})}{\alpha_i \preceq_{mt} \alpha_j}$$

$$\text{(FORK)}\ \frac{\alpha_i = \texttt{fork}(\texttt{t}, \texttt{t}') \qquad \alpha_j = \texttt{threadinit}(\texttt{t}')}{\alpha_i \preceq_{mt} \alpha_j}$$

$$\text{(JOIN)}\ \frac{\alpha_i = \texttt{threadexit}(\texttt{t}') \qquad \alpha_j = \texttt{join}(\texttt{t}, \texttt{t}')}{\alpha_i \preceq_{mt} \alpha_j}$$

$$\text{(LOCK)}\ \frac{\alpha_i = \texttt{release}(\texttt{t}, \texttt{l}) \qquad \alpha_j = \texttt{acquire}(\texttt{t}', \texttt{l})}{\alpha_i \preceq_{mt} \alpha_j}$$

$$\text{(TRANS-MT)}\ \frac{\alpha_i \preceq \alpha_k \qquad \alpha_k \preceq \alpha_j}{\alpha_i \preceq_{mt} \alpha_j}$$

Execution Trace

Race Detection for Event-Driven Mobile Applications, *PLDI 14*

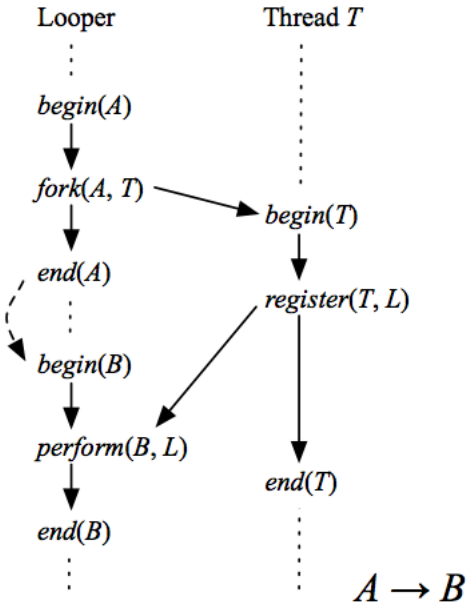
- begin(t), end(t)
- rd (t, x), wr (t, x)
- fork(t,u), join(t,u).
- wait(t,m), notify(t,m)

- ***send(t,e,delay):*** enqueues a new event e at the end of the event queue in task t. e would be executed after *delay*
- ***sendAtFront (t, e):*** enqueues a new event e at the front of the event queue in task t.
- ***register (t, l)*** and ***perform (t, l)***: models the *event listener* programming construct in Android.

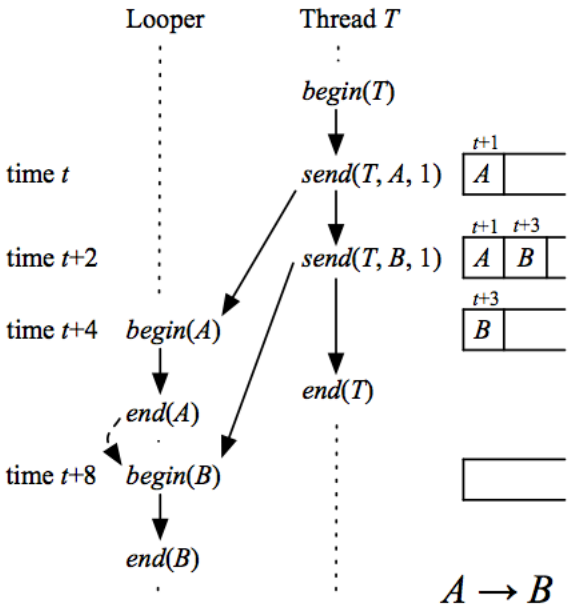
Happens-before rule

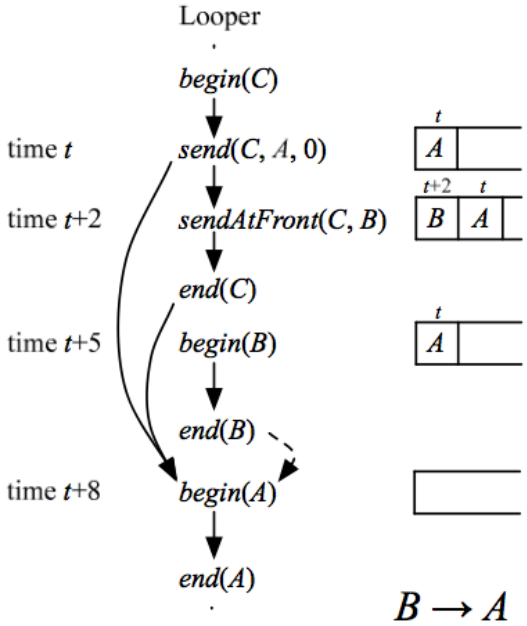
Race Detection for Event-Driven Mobile Applications, *PLDI 14*

- Causality model:
  - Conventional happens-before relations:
    - Program-order rule; Fork-join rule; Signal-and-wait rule
  - Event generation and execution:
    - Event listener rule; Send rule; External input rule
  - Atomicity rule
  - Event queue rule



Atomicity rule: If $begin(e_1) \prec_\alpha end(e_2)$, then $end(e_1) \prec_\alpha begin(e_2)$.



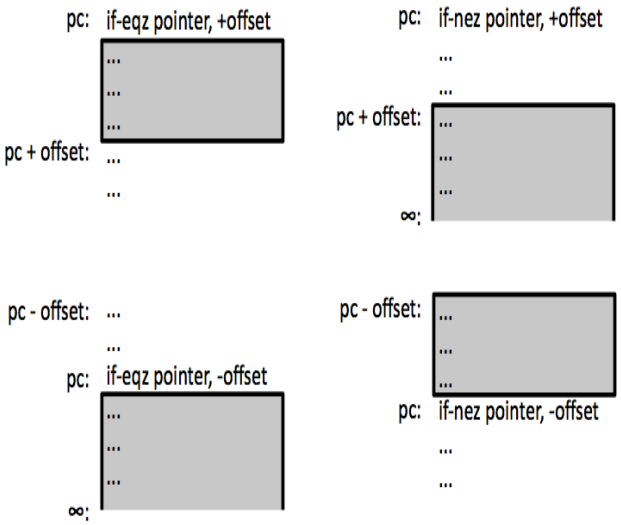


1. If $send(t_1, e_1, delay_1) \prec_\alpha send(t_2, e_2, delay_2)$ and $delay_1 \leq delay_2$, then $end(e_1) \prec_\alpha begin(e_2)$.
2. If $send(t_1, e_1, delay_1) \prec_\alpha sendAtFront(t_2, e_2)$ and $sendAtFront(t_2, e_2) \prec_\alpha begin(e_1)$, then $end(e_2) \prec_\alpha begin(e_1)$.
3. If $sendAtFront(t_1, e_1) \prec_\alpha send(t_2, e_2, delay_2)$, then $end(e_1) \prec_\alpha begin(e_2)$.
4. If $sendAtFront(t_1, e_1) \prec_\alpha sendAtFront(t_2, e_2)$ and $sendAtFront(t_2, e_2) \prec_\alpha begin(e_1$, then $end(e_2) \prec_\alpha begin(e_1)$.

False Positives elimination

Race Detection for Event-Driven Mobile Applications, *PLDI 14*

- If-Guard check: programmers often check if a pointer is null before using it.
  - if-eqz (jump if a pointer is null)
  - if-nez (jump if a pointer is not null)
  - if-eq (jump if two pointers are equal)



- Intra-event-allocation
  - if there is an allocation after a free in an event
  - if there is an allocation before a use within the same event

Remaining

- Implementation:
  - Instrument Android ROM
    - Dalvik Virtual Machine (java.lang.Object.wait, synchronized{}, …)
    - Java Core libraries (java.lang.thread, …)
    - Android Core libraries (android.os.Handler, android.os.Looper)
    - Binder IPC Framework
  - Offline analysis tool
    - Algorithm generating DAG
    - Reachability algorithm for operations on every single memory

Remaining

- More:
  - Add If-Guard check and Intra-event-allocation check
- Evaluation:
  - F-Droid open-source applications
  - My application for research

Problems

- Instrumentation Approach
  - DroidRacer
- Offline running time
  - varies from 30 minutes to a whole day

Thanks!

Questions