Reconfiguration Planning

CS 598D, Spring 2010
Princeton University

April 19, 2010

Sanjai Narain
narain@research.telcordia.com
908 337 3636

Outline

- Given an initial and final configuration, all components cannot be concurrently reconfigured
- Problem: In what order should the components be reconfigured so an invariant is never falsified during transition?
  - Invariant examples: security policy not violated, mission-critical services not disrupted
- Related problems
  - How to synthesize only reachable final configurations?
  - How to let configuration variables assume intermediate values?
- Solution
  - Transform an invariant into a constraint on times at which configurations are changed and solve for the times
  - Strengthen synthesis constraint with this to ensure that final configuration is reachable
  - Generalize to allow parameters to assume intermediate values

Example 1: Encryption Before Routing

- route and tunnel are two variables on a router, representing presence of a static route and an IPSec tunnel, respectively

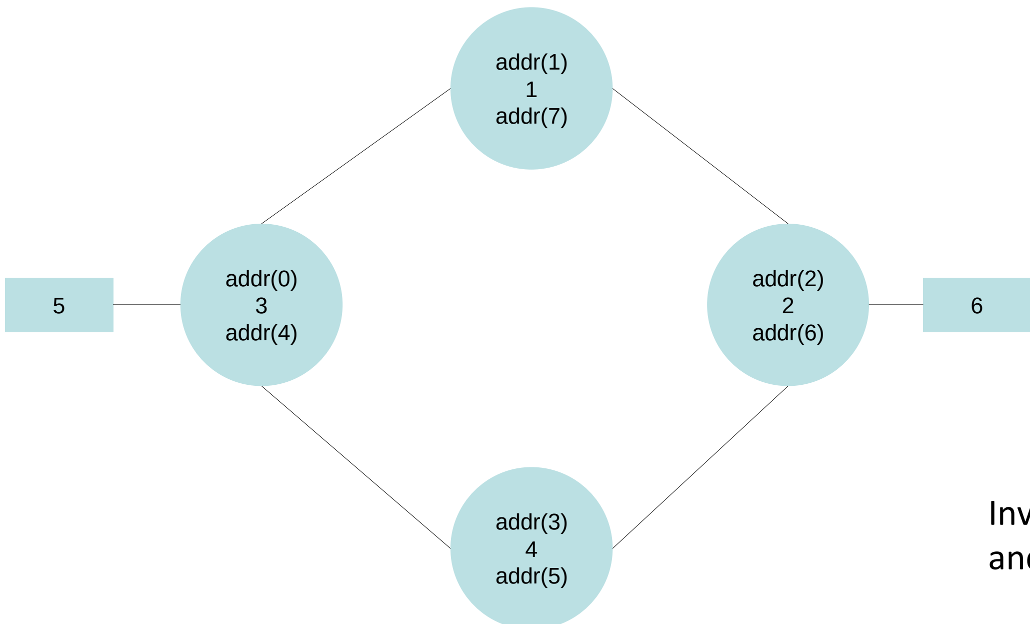
- init[route] = 0; final[route]=1

- init[route] = 0; final[route] = 1

- Invariant: Data only leaves router when encrypted
  - route=1 => tunnel=1

- Safe reconfiguration plan is [tunnel, route] but not [route, tunnel]

Safely Decommissioning Router 1



Next hop variable	Initial value	Final value
addr(0)	1	4
addr(1)	2	0
addr(2)	6	6
addr(3)	0	2
addr(4)	5	5
addr(5)	0	3
addr(6)	1	4
addr(7)	3	0

Invariant: Maintain bidirectional connectivity
and(or(routing_5_6_via_1, routing_5_6_via_4),
    or(routing_6_5_via_1, routing_6_5_via_4))

routing_5_6_via_1 = and(addr(0)=1, addr(1)=2, addr(2)=6)
routing_5_6_via_4 = and(addr(0)=4, addr(3)=2, addr(2)=6)
routing_6_5_via_1 = and(addr(6)=1, addr(7)=3, addr(4)=5)
routing_6_5_via_4 = and(addr(6)=4, addr(5)=3, addr(4)=5)

Reconfiguration plan: [addr(2), addr(5), addr(4), addr(3), addr(0), addr(1), addr(6), addr(7)]

Reconfiguration Planning Algorithm

- Define Invariant to be preserved as a quantifier free form. This is a Boolean combination of:
  - x op y
  - contained(a, m, b, n)

  where x, y, a, m, b, n are integer variables or constants and op in {=,<,>,<=,>=}

- For each configuration variable v in Invariant, define a new variable time[v] at which v changes from init[v] to final[v]. Distinct variables change at distinct times

- holds[t] = result of replacing each variable v in Invariant by if time[v] =< t then init[v] else final[v]

- holds_all_times = conjunction of holds[1],..,holds[k] where k is the number of configuration variables

- Solve holds_all_times to find time[v] for each v

- For Example 1:
  - Invariant is (route=1 => tunnel=1)

  - holds[t]= (if time[route]=<t then 1 else 0)=1 => (if time[tunnel]=<t then 1 else 0)=1

  - Solving holds_all_times produces the solution: time[tunnel]=1, time[route]=2

  - The reconfiguration plan is then [tunnel, route]

Synthesizing Reachable Final Configuration

- H drops packets from G whose size is larger than H's MTU, and whose Do Not Fragment bit is set
- H also sends warning to G in an ICMP message so G can reduce the size of transmitted packets
- However, G may block ICMP so G will continue to send large packets that H will drop
- Initial state:
  - MTU at both routers is 1500 and ICMP is blocked.
- Requirement for final state:
  - MTUs of both routers is 1600
- Solution: MTUs of both routers is 1600 and ICMP is blocked
- But if Invariant is that there is no packet loss due to MTU mismatch then this final state is not reachable
- A reachable final state is one where the MTU is 1600 and ICMP is enabled for both routers.
- To compute reachable final state, for each variable v, let final[v]=v. Now solve for v and time[v]

Synthesizing Reachable Final Configuration contd.

- Req = and(gmtu = 1600, hmtu = 1600)
- Invariant = or(gmtu = hmtu, and(gicmp = 1, hicmp = 1))
- The final values of variables are:
  - gmtu= 1600
  - hmtu= 1600
  - gicmp= 1
  - hicmp= 1
- The reconfiguration plan is [gicmp, hicmp, gmtu, hmtu].
  - First enable ICMP at both routers and then increase the MTU

Allowing Variables To Assume Intermediate Values

- Motivating example: Swap distinct IP addresses without introducing duplicates
- In addition to taking on init[v] and final[v], let variable v also take on a single intermediate value mid[v].
- v changes to mid[v] at time t1[v] and to final[v] at time t2[v].
- To compute holds[t], replace every occurrence of v in Invariant by:

  if t1[v]<=t then init[v]
  else if t2[v]<=t then mid[v]
  else final[v]
- holds_all_times is the conjunction of holds[t] for each value of t in 1,..,2*k where k is the number of configuration variables.
- mid[v] need not be known in advance. It is treated as another configuration variable whose value, along with those of t1[v] and t2[v] will be computed when holds_all_times is solved.
- This idea is generalized to the multiple intermediate values

Putting The Pieces Together

- Given an initial configuration, a requirement on a final state, and an invariant, one can compute a final configuration that:
  - Satisfies requirement
  - Is minimum cost distance from initial configuration
  - Is reachable while preserving invariant

- Constraint solving unifies solutions to all these problems