Modelling and Verification of Protocols for Wireless Networks

(Lecture 5)

Peter Höfner

(Lecture at University of Twente, Jan/Feb 2017)

www.data61.csiro.au

last update: Jan 31, 2017

Admin - Next Week

- Monday: Last Lecture: Q&A and open problems
- Monday afternoon: last lab - discussion of individual projects
- Tuesday morning: Oral exams in Zi 3063

Contents of this Lecture

What should you have learnt

- Formalising Properties
- Invariants
  - local vs global properties
- Reachability
  - problems
  - progress
  - fairness and its problems

Invariants

Invariants

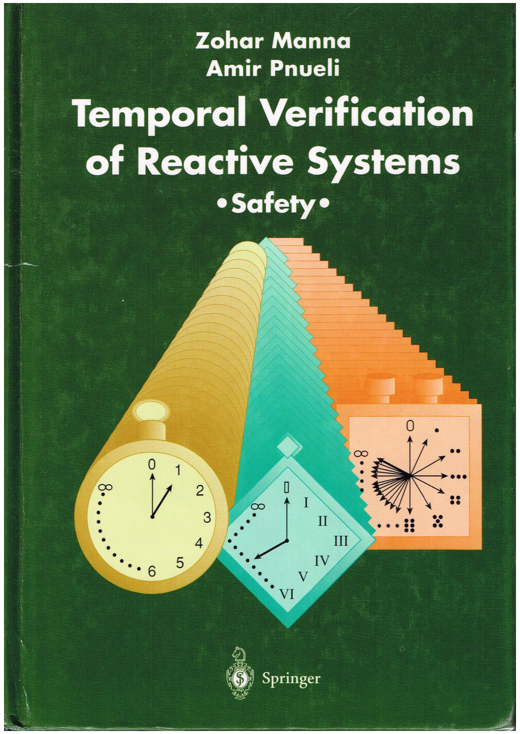
For an assertion φ,

$$\begin{array}{ll} \text{B1.} & \Theta \rightarrow \varphi \\ \text{B2.} & \{\varphi\}\ \mathcal{T}\ \{\varphi\} \\ \hline & \Box\,\varphi \end{array}$$

Fig. 1.1. Rule INV-B (basic invariance).

statement true for initial state

show for a single step



Auxiliary Invariants for AODV

- All routing table entries have a hop count greater or equal than 1.

$$(*,*,*,*,hops,*,*) \in \xi_N^{ip}(\mathtt{rt}) \;\Rightarrow\; hops \geq 1$$

just some decoration to identify node, and state of the network

Auxiliary Invariants

- Easy to encode (pen and paper; Uppaal; Isabelle/HOL)
- only concern sequential processes (local state)

Loop Freedom

- The quality of the routing table entries for a destination dip is strictly increasing along a route towards *dip*, until it reaches either *dip* or a node with an invalid routing table entry to *dip*.

$$dip \in \mathtt{vD}_N^{ip} \cap \mathtt{vD}_N^{nhip} \wedge nhip \neq dip \;\Rightarrow\; \xi_N^{ip}(\mathtt{rt}) \sqsubset_{dip} \xi_N^{nhip}(\mathtt{rt})$$

- property of networks (not sequential process any more)
- but ξ, the evaluation function, is locally defined for seq. processes

Loop Freedom - Encoding

- Pen-and-Paper analysis
  - easy: just make up new notation (as we did)

Reachability Properties (based on LTL or CTL)

CTL - Recap

- interpreted over transition systems
  (that can be derived from the SOS rules of AWN)

- built from atomic propositions, e.g. “two nodes are connected”
- Synta›
  - True is in CTL
  - $\boldsymbol{x} \in \mathcal{P}$ is in CTL
  - $\boldsymbol{P}, \boldsymbol{Q} \in$ CTL, $\neg \boldsymbol{P} \in$ CTL and $\boldsymbol{P} \wedge \boldsymbol{Q} \in$ CTL
  - E X φ is in CTL if φ is in CTL
  - A X φ is in CTL if φ is in CTL
  - A (φ_1 UNTIL φ_2) is in CTL if $\varphi_1, \varphi_2 \in$ CTL,
  - E (φ_1 UNTIL φ_2) is in CTL if $\varphi_1, \varphi_2 \in$ CTL,

LTL - Recap

- evaluated on paths, so it does not requires A- and E-operators

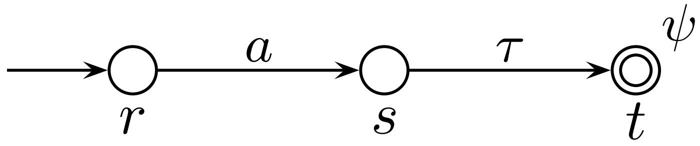
Leads-To

- $\mathbf{A}\,\mathbf{G}(\varphi^{pre} \;\Rightarrow\; \mathbf{A}\,\mathbf{F}\varphi^{post})$

- sometimes a side condition, which will hold "constantly" should be added

$$\mathbf{A}\,\mathbf{G}(\varphi^{pre} \;\Rightarrow\; \mathbf{A}\,\mathbf{F}(\varphi^{post} \vee \neg\psi))$$

Progress



- system stops in state *s* without ever forming an internal transition

- $\mathbf{F}\psi$ should be true when in state *s*, but not necessarily in state *a*, e.g., when *a* is a receive action

A process in a state that admits an internal transition τ or and output transition will eventually perform a transition.

Output Transitions

- sometimes also called output actions
  (since the output transitions are completely determined by the actions).

- in AWN, on the layer of entire networks, assignments, guards, etc. were encoded as internal actions, so the only output transitions are

$$ip : \mathbf{deliver}(d)$$

e

$$R :\ *\mathbf{cast}(m)$$

$$\mathbf{broadcast}(m),\ \mathbf{groupcast}(D,m),\ \mathbf{unicast}(dip,m), \neg\mathbf{unicast}(dip,m)$$

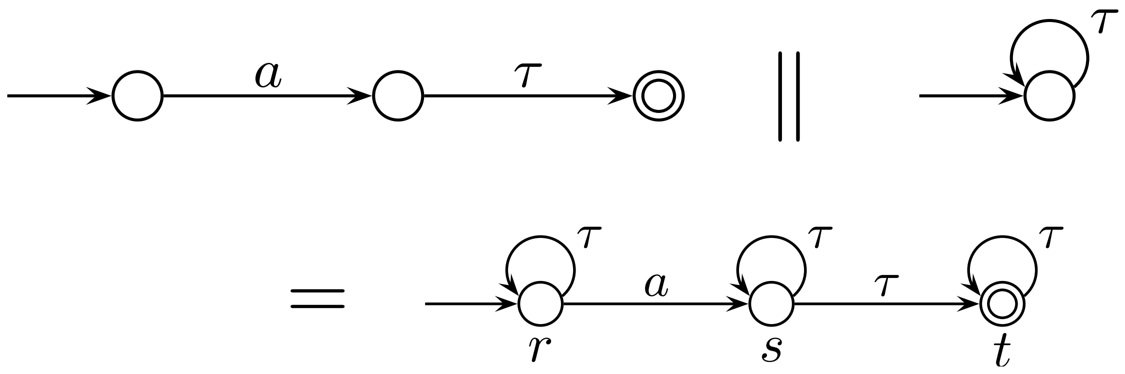
Progress in Uppaal and other formalisms

- Uppaal
  - use of committed/urgent states
  - use of invariants (in the timed model)

- early work on temporal logics considered only infinite (and unlabelled) paths

- more general: consider *complete paths*
  - a path to be *complete* iff it is either infinite or ends in a state from which no further internal or output transitions are possible
  - properties are satisfied iff they hold on all complete paths

Justness



- does $\mathbf{G}(a \;\Rightarrow\; \mathbf{F}(\psi))$ hold?

A component in a parallel composition in a state that admits an internal or output transition will eventually perform a transition.

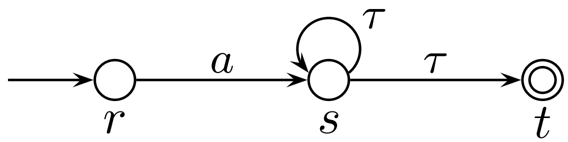
Justness in AWN

- defined via the entire hierarchy of layers
- avoids the existence of premature paths
  - a path starting from any AWN expression (i.e. a sequential or parallel process expression, a node expression or (partial) network expression) ends prematurely if it is finite and from its last state an internal or output transitions is possible.
  - refinement of the definition of a complete path

- using this definition $\mathbf{G}(a \;\Rightarrow\; \mathbf{F}(\psi))$ holds

- Justness in Uppaal: I do not know

Fairness



- $\mathbf{G}(a \;\Rightarrow\; \mathbf{F}(\psi))$ does not hold under justness
- often global fairness assumptions are made
  - weak: if a transition is enabled continuously, it will be taken infinitely often

$$\mathbf{F}\,\mathbf{G}(enabled(a)) \;\Rightarrow\; \mathbf{G}\,\mathbf{F}(a)$$

  - strong: if a transition is enable infinitely often, it will be taken infinitely often

$$\mathbf{G}\,\mathbf{F}(enabled(a)) \;\Rightarrow\; \mathbf{G}\,\mathbf{F}(a)$$

Fairness may be too strong

- non-deterministic choice may always choose “the other” transition
- in AWN, we consider all occurrences of choice, and decide individually
  - e.g. choice in the main process (Lines 21 ,33)
    here we postulate a weak fairness assumption

Why Fairness is dangerous

- Consider the following two programs

$$x := 1 \quad \| \quad \textbf{repeat} \;\; y := y + 1 \;\; \textbf{forever}$$

$$\begin{array}{l}
\textbf{repeat} \\
\quad \textbf{case} \\
\quad\quad \textbf{if} \;\; \texttt{True} \;\; \textbf{then} \;\; \text{y:=y+1} \;\; \textbf{fi} \\
\quad\quad \textbf{if} \;\; x = 0 \;\; \textbf{then} \;\; \text{x:=1} \;\; \textbf{fi} \\
\quad \textbf{end} \\
\textbf{forever}
\end{array}$$

- side remark: these programs are bisimilar

Main Process (cont'd)

```
+ [ Let dip ∈ qD(store) ∩ vD(rt) ]          /* send a queued data packet if a valid route is known */
  [[data := head(σ_queue(store,dip))]]
   unicast(nhop(rt,dip),pkt(data,dip,ip)) .
     [[store := drop(dip,store)]]          /* drop data from the store for dip if the transmission was successful */
     AODV(ip,sn,rt,rreqs,store)
   ▶ /* an error is produced and the routing table is updated */
     [[dests := {(rip,inc(sqn(rt,rip))) | rip ∈ vD(rt) ∧ nhop(rt,rip) = nhop(rt,dip)}]]
     [[rt := invalidate(rt,dests)]]
     [[store := setRRF(store,dests)]]
     [[pre := ⋃{precs(rt,rip) | (rip,∗) ∈ dests}]]
     [[dests := {(rip,rsn) | (rip,rsn) ∈ dests ∧ precs(rt,rip) ≠ ∅}]]
     groupcast(pre,rerr(dests,ip)) . AODV(ip,sn,rt,rreqs,store)
+ [ Let dip ∈ qD(store) − vD(rt) ∧ σ_p-flag(store,dip) = req ]          /* a route discovery process is initiated */
  [[store := unsetRRF(store,dip)]]          /* set request-required flag to no-req */
  [[sn := inc(sn)]]          /* increment own sequence number */
  /* update rreqs by adding (ip,nrreqid(rreqs,ip)) */
  [[rreqid := nrreqid(rreqs,ip)]]
  [[rreqs := rreqs ∪ {(ip,rreqid)}]]
  broadcast(rreq(0,rreqid,dip,sqn(rt,dip),sqnf(rt,dip),ip,sn,ip)) . AODV(ip,sn,rt,rreqs,store)
```

Fairness Assumptions for AODV

- whenever the node *ip* perpetually has queued packets for *dip* as well as a (valid) route to *dip*, it will eventually forward a data packet originating from *ip* towards *dip*

$$\mathbf{G}\Big(\mathbf{G}(dip \in \mathtt{qD}^{ip} \cap \mathtt{vD}^{ip}) \;\Rightarrow\; \mathbf{F}\big(\mathbf{unicast}(*,\mathtt{pkt}(*,dip,ip))\big)\Big)$$

- whenever *ip* perpetually has queued packets for *dip* but no valid route to *dip*, then node *ip* does issue a request for a route from *ip* to *dip*
(is also considers a request-required flag)

$$\mathbf{G}\Big(\mathbf{G}(dip \in \mathtt{qD}^{ip} - \mathtt{vD}^{ip} \;\wedge\; \sigma^{ip}_{p\text{-}flag}(dip) = \mathtt{req}) \;\Rightarrow\; \mathbf{F}\big(\mathbf{broadcast}(\mathtt{rreq}(*,*,dip,*,*,ip,*,ip))\big)\Big)$$

- there is no need to formalise a fairness assumption for Line 1 — Why?

Fairness Assumption for QMSG

- whenever there is a stored message, it will be passed on

$$\mathbf{G}\big(\mathbf{G}(\mathtt{msgs}^{ip} \neq [\,]) \;\Rightarrow\; \mathbf{F}\big(ip : \mathbf{send}(*)\big)\big)$$

Properties for AODV

Route Discovery

$$\mathbf{G}\Big(\big(\mathbf{connected}^*(oip,dip) \wedge \mathbf{broadcast}(\mathtt{rreq}(*,*,dip,*,*,oip,*,oip))\big) \Rightarrow \mathbf{F}\big(dip \in \mathtt{vD}^{oip} \vee \mathbf{disconnect}(*,*)\big)\Big)$$

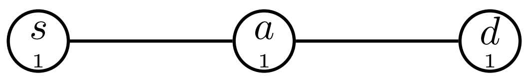
(multihop) connection between oil and dip

ANY link breaks

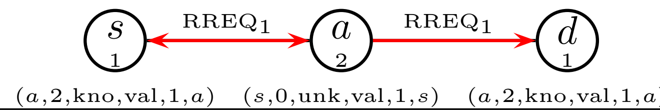
- route discovery does not hold
- can easily be fixed

Example of Failure

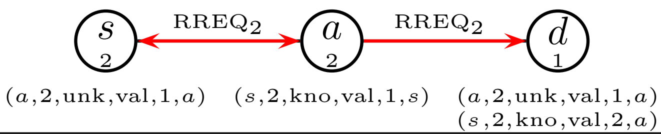
(a) The initial state.



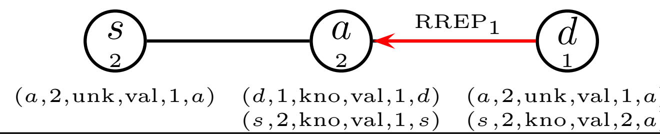
(b) a broadcasts a new RREQ message destined to d; all nodes receive the RREQ and update their RTs.



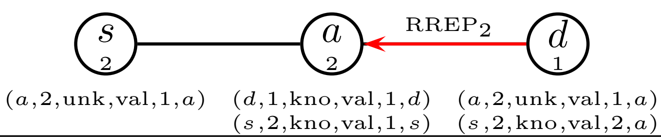
(c) s broadcasts a new RREQ destined to d; a forwards it.



(d) d handles RREQ$_1$ and unicasts a RREP to a.



(e) d handles RREQ$_2$ and unicasts a RREP to a.



(f) This ends the work of AODV; s will never get an answer for its RREQ.

Packet Delivery (1)

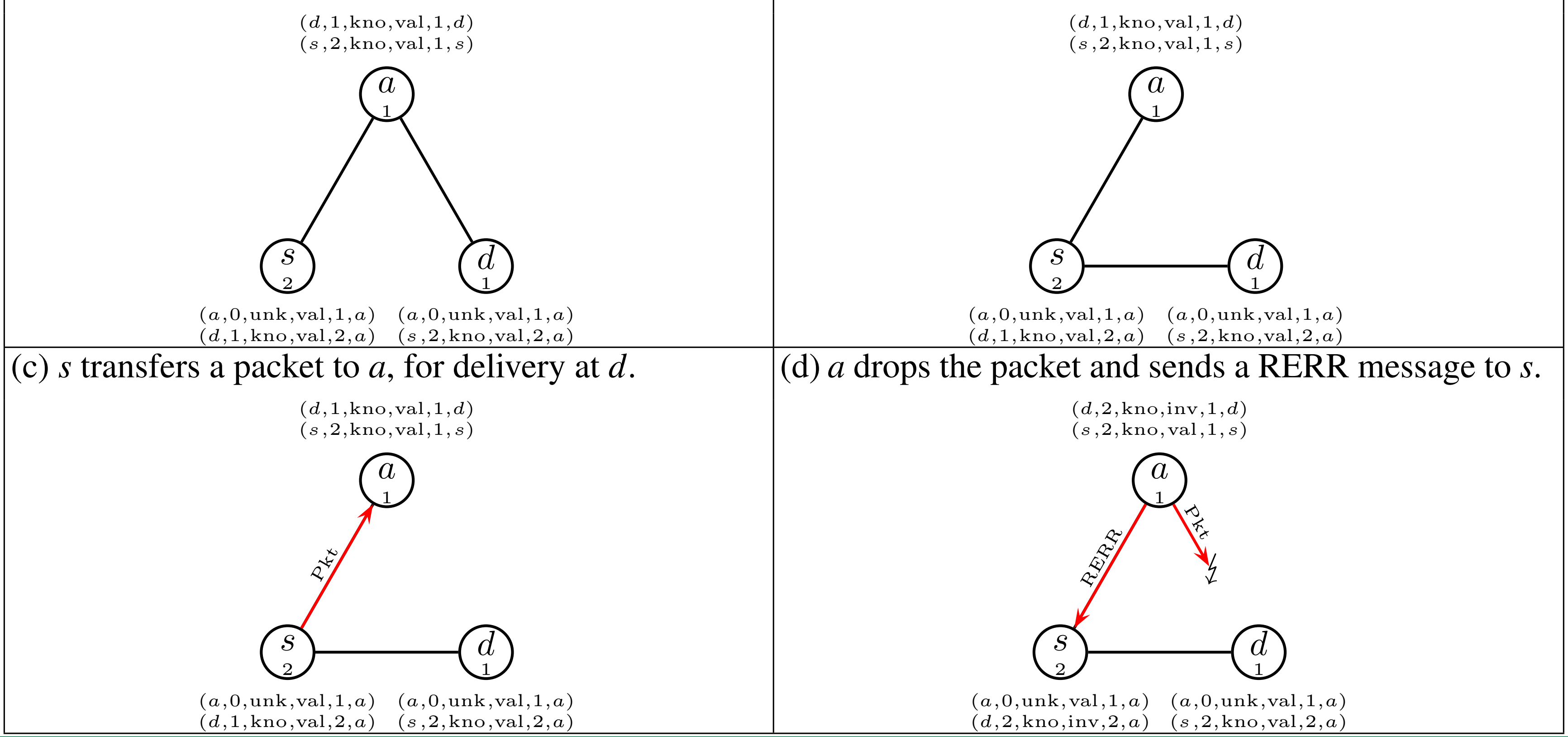
$$\mathbf{G}\left(\begin{array}{c}\left(\mathbf{connected}^*(oip,dip)\ \wedge\ oip:\mathbf{newpkt}(dp,dip)\right)\\ \Rightarrow\ \mathbf{F}\left(dip:\mathbf{deliver}(dp)\ \vee\ \mathbf{disconnect}(*,*)\right)\end{array}\right)$$

- route deliver does not hold (even with the fixed route discovery algorithm)
- but this is normal behaviour of any routing protocol for wireless networks!

Example of Failure

$$(dip, sqn, k, v, hc, nhip)$$

- destination
- sequence number (freshness)
- known/ unknown
- validity
- hop count
- next hop



(a) The initial state; s has established a route to d.

(b) The topology changes.

(c) s transfers a packet to a, for delivery at d.

(d) a drops the packet and sends a RERR message to s.

Packet Delivery (2)

$$\mathbf{G}\left(\begin{array}{c}\big(\mathbf{connected}^*(oip,dip)\ \wedge\ \mathbf{GF}\big(oip:\mathbf{newpkt}(dp,dip)\big)\big)\\ \Rightarrow\ \mathbf{F}\big(dip:\mathbf{deliver}(dp)\ \vee\ \mathbf{disconnect}(*,*)\big)\end{array}\right)$$

- add side condition: $\psi = \mathbf{F}\big(oip:\mathbf{newpkt}(dp,dip)\big)$
  (keep injecting the same packet again and again)

- seems to be reasonable formalisation for a routing protocol;
- still it is too strong for AODV (a flow!)
  (there is a problem with the flag)

Packet Delivery (3)

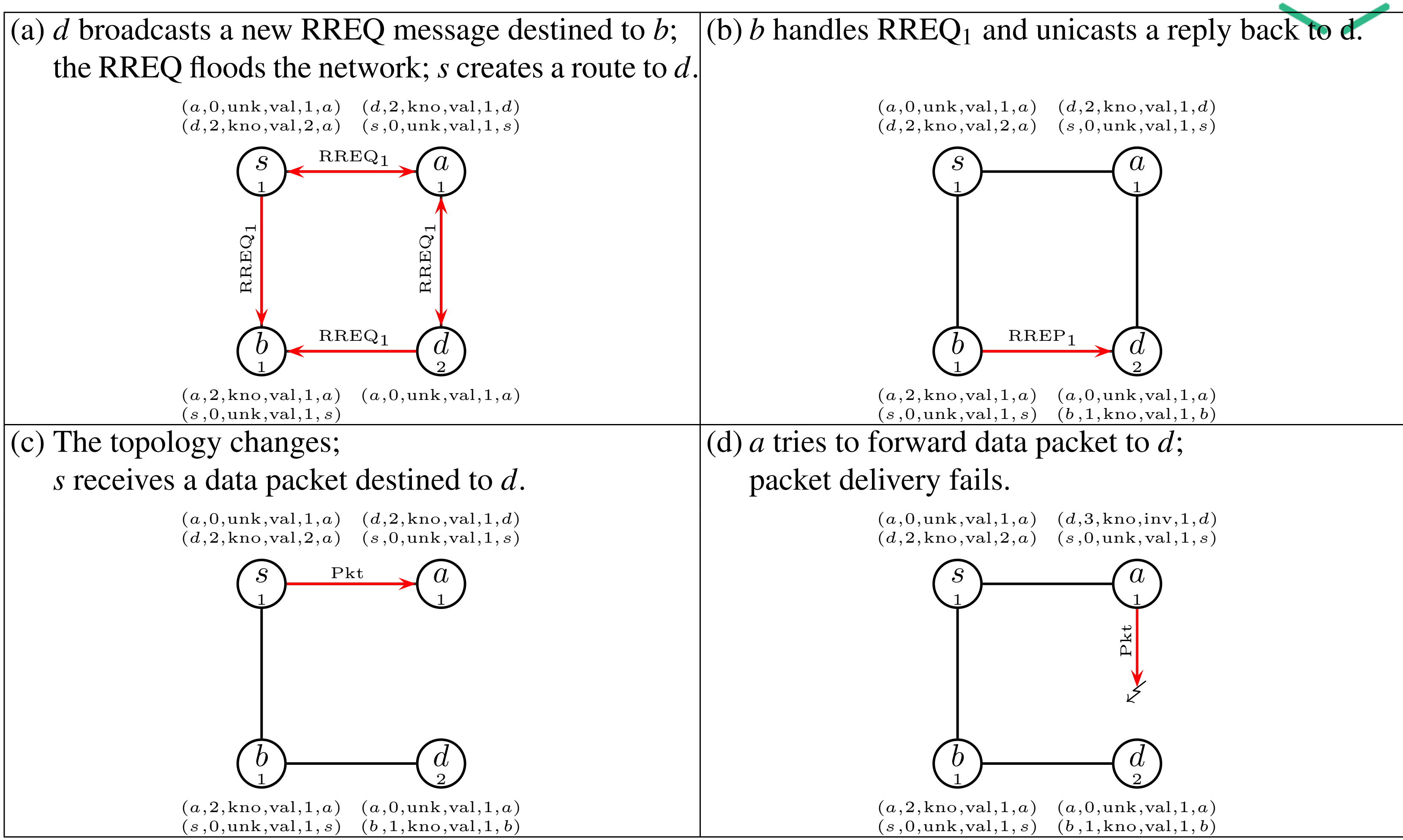
$$\mathbf{G}\big(\mathbf{G}(dip \in \mathtt{qD}^{oip} - \mathtt{vD}^{oip}) \;\Rightarrow\; \mathbf{F}\big(\sigma^{oip}_{p\text{-}flag}(dip) = \mathtt{req}\big)$$

$$\Rightarrow\; \mathbf{G}\left(\begin{array}{c} \mathbf{connected}^{*}(oip, dip) \\ \Rightarrow\; \mathbf{F}\big(dip : \mathbf{deliver}(dp) \;\vee\; \mathbf{disconnect}(*,*) \;\vee\; \neg\mathbf{F}\big(oip : \mathbf{newpkt}(dp, dip)\big)\big) \end{array} \right)$$

- add another side condition: $\mathbf{G}\big(\mathbf{G}(dip \in \mathtt{qD}^{oip} - \mathtt{vD}^{oip}) \;\Rightarrow\; \mathbf{F}(\sigma^{oip}_{p\text{-}flag}(dip) = \mathtt{req})\big)$
  (keep injecting the same packet again and again)

- seems (even more) to be a reasonable formalisation for a routing protocol;
- still AODV does not satisfy this property either
  (now the problem lies in *precursor lists;* these lists contain neighbours interested in)

Example of Failure



Summary

- Invariants
  - often depend on local data structure
  - hence system dependent
  - for automatic analysis local data structure can be complicated

- Reachability
  - problems with progress and fairness
    (needs careful decisions)
  - (weak/strong) fairness is often too strong
  - properties should be (more or less) independent of the protocol
    - are they?

References

- A. Fehnker, R.J. van Glabbeek, P. Höfner, M. Portmann, A. McIver and W.L. Tan: *A Process Algebra for Wireless Mesh Networks used for Modelling, Verifying and Analysing AODV.* Technical Report 5513, NICTA. 2013. arXiv: CoRR abs/1312.7645
- R.J. van Glabbeek and P. Höfner: *Progress, Fairness and Justness in Process Algebra.* arXiv: CoRR abs/1501.03268
- Z. Manna and A. Pnueli: *Temporal Verification of Reactive Systems - Safety.* Springer, 1995