



Using Process Algebra to Design Better Protocols

Peter Höfner
January 2017

www.data61.csiro.au



Why Better Protocols are Needed



- Routing Protocols are Broken
 - Routing Protocols establish **non-optimal routes**
 - AODV Routing Protocol sends packets in **loops**
 - Chord Protocol is **not correct**
 - BGP **oscillates** persistent routes
 - ...

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

Computer Networks 32 (2000) 1–16

Persistent route oscillations in inter-domain routing[☆]

Kannan Varadhan^{a,*}, Ramesh Govindan^b, Deborah Estrin^b

^a Lucent Technologies, Room MH 2B-230, 600 Mountain Avenue, Murray Hill, NJ 07974, USA

^b USC Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90292, USA

^c New South Wales, Australia
Peter.Hoefner@nicta.com.au

Why the Chord Ring-Mesh Networks:
Maintenance
Is Not Correct (Extended Abstract)

Routing Decisions
AT&T Laboratories—Research, Florham Park, New Jersey, USA
Pamela Zave
Email: pamela@research.att.com
Stanislaw...

Today's Protocol Development



- IETF: “Rough Consensus and Running Code” (Trial and Error)
 - start with a good idea
 - build a protocol out of it (implementation)
 - run tests (over several years)
 - find limitations, flaws, etc...
 - fix problems
 - build a new version of the protocol
 - at some point people agree on an RFC (request for comments)



Beauvais Cathedral
(~300 years to build, at least 2 collapses)

Better Protocols are Needed Now!



- We cannot afford this approach
 - to expensive w.r.t. time
 - to expensive w.r.t. money
 - we are not working in a lab, i.e., sometimes we have one try only (e.g. BGP)
- Is there a method which is more reliable and cost efficient



The original design was so boldly conceived that it was found structurally impossible to build.

What's the Problem? (1)



- Specifications are (excessively) long
 - the Session Initiation Protocol is 268 pages long (and not even self contained - by 2009 142 additional documents were required)
 - IEEE 802.11 is 2.793 pages long



What's the Problem? (2)



- Specifications are
 - underspecified
 - contradictory
 - erroneous, and
 - ambiguous

What's the Problem? (3)

- Specifications are written in English Prose
 - in case of AODV there are 5 different implementations, all compliant to the standard



Aims



- Provide complete and practical formal methods
 - expressive
(mobility, dynamic topology, types of communication,...)
 - usable and intuitive
 - description language + proof methodology + automation
- Specification, verification and analysis of protocols
 - formalise relevant standard protocols
 - analyse the protocols w.r.t. key requirements
 - analyse compliant implementations
- Development of improved protocols
 - assured protocol correctness
 - improve reliability and performance

Developed Process Algebra



- Description Language (Syntax)

$X(\text{exp}_1, \dots, \text{exp}_n)$	process calls
$P + Q$	nondeterministic
$[\varphi]P$	if-construct (guard)
$\llbracket \text{var} := \text{exp} \rrbracket P$	assignment followed
broadcast $(ms).P$	broadcast
groupcast $(\text{dests}, ms).P$	groupcast
unicast $(\text{dest}, ms).P \blacktriangleright Q$	unicast
send $(ms).P$	send
receive $(\text{msg}).P$	receive
deliver $(\text{data}).P$	deliver

Developed Process Algebra



- Description Language (Syntax)

$[\varphi]P + [\neg\varphi]Q$	deterministic choice
$P(n) = \llbracket n := n + 1 \rrbracket.P(n)$	loops

- Do we need more?

$P \ll Q$	parallel operator on nodes
-----------	----------------------------

Case Study: AODV



```
+ [ (oip, rreqid) ∉ rreqs ]      /* the RREQ is new to this node */
  [[rt := update(rt, (oip, osn, kno, val, hops + 1, sip, 0))]      /* update the route to oip in rt */
  [[rreqs := rreqs ∪ {(oip, rreqid)}]]      /* update rreqs by adding (oip, rreqid) */
  (
    [ dip = ip ]      /* this node is the destination node */
    [[sn := max(sn, dsq)]      /* update the sqn of ip */
    /* unicast a RREP towards oip of the RREQ */
    unicast(nhop(rt, oip), rrep(0, dip, sn, oip, ip)) . AODV(ip, sn, rt, rreqs, store)
    ▶ /* If the transmission is unsuccessful, a RERR message is generated */
    [[dests := {(rip, inc(sqn(rt, rip))) | rip ∈ vD(rt) ∧ nhop(rt, rip) = nhop(rt, oip)}]]
    [[rt := invalidate(rt, dests)]]
    [[store := setRRF(store, dests)]]
    [[pre := ∪ {precs(rt, rip) | (rip, *) ∈ dests}]]
    [[dests := {(rip, rsn) | (rip, rsn) ∈ dests ∧ precs(rt, rip) ≠ 0}]]
    groupcast(pre, rerr(dests, ip)) . AODV(ip, sn, rt, rreqs, store)
  + [ dip ≠ ip ]      /* this node is not the destination node */
    (
      [ dip ∈ vD(rt) ∧ dsq ≤ sqn(rt, dip) ∧ sqnf(rt, dip) = kno ]      /* valid route to dip that is fresh enough */
      /* update rt by adding precursors */
      [[rt := addpreRT(rt, dip, {sip})]]
      [[rt := addpreRT(rt, oip, {nhop(rt, dip)})]]
      /* unicast a RREP towards the oip of the RREQ */
      unicast(nhop(rt, oip), rrep(dhops(rt, dip), dip, sqn(rt, dip), oip, ip)) .
```

Developed Process Algebra



- Semantics
 - not used by a software engineer
 - internal state determined by expression and valuation

$$\begin{aligned} \xi, \mathbf{broadcast}(ms).p & \xrightarrow{\mathbf{broadcast}(\xi(ms))} \xi, p \\ \xi, \mathbf{groupcast}(dests, ms).p & \xrightarrow{\mathbf{groupcast}(\xi(dests), \xi(ms))} \xi, p \\ \xi, \mathbf{unicast}(dest, ms).p \blacktriangleright q & \xrightarrow{\mathbf{unicast}(\xi(dest), \xi(ms))} \xi, p \\ \xi, \mathbf{unicast}(dest, ms).p \blacktriangleright q & \xrightarrow{\neg \mathbf{unicast}(\xi(dest))} \xi, q \\ \xi, \mathbf{send}(ms).p & \xrightarrow{\mathbf{send}(\xi(ms))} \xi, p \\ \xi, \mathbf{deliver}(data).p & \xrightarrow{\mathbf{deliver}(\xi(data))} \xi, p \\ \xi, \mathbf{receive}(msg).p & \xrightarrow{\mathbf{receive}(m)} \xi[msg := m], p \quad (\forall m \in \text{MSG}) \end{aligned}$$

Developed Process Algebra



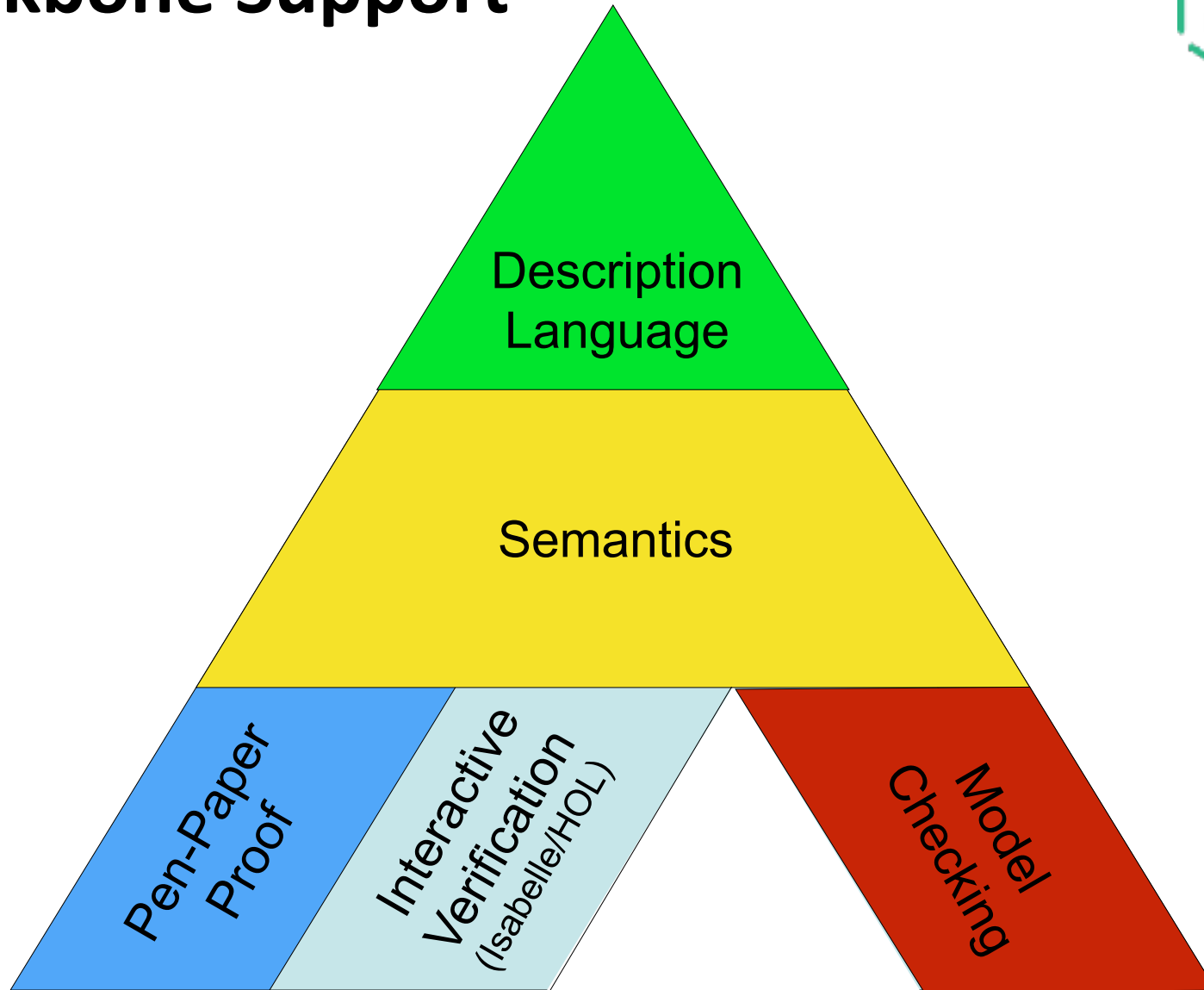
- Semantics cont'd

$$\frac{P \xrightarrow{a} P'}{P \ll Q \xrightarrow{a} P' \ll Q} \quad (\forall a \neq \mathbf{receive}(m))$$

$$\frac{Q \xrightarrow{a} Q'}{P \ll Q \xrightarrow{a} P \ll Q'} \quad (\forall a \neq \mathbf{send}(m))$$

$$\frac{P \xrightarrow{\mathbf{receive}(m)} P' \quad Q \xrightarrow{\mathbf{send}(m)} Q'}{P \ll Q \xrightarrow{\tau} P' \ll Q'} \quad (\forall m \in \mathbf{MSG})$$

Backbone Support



Case Study: AODV



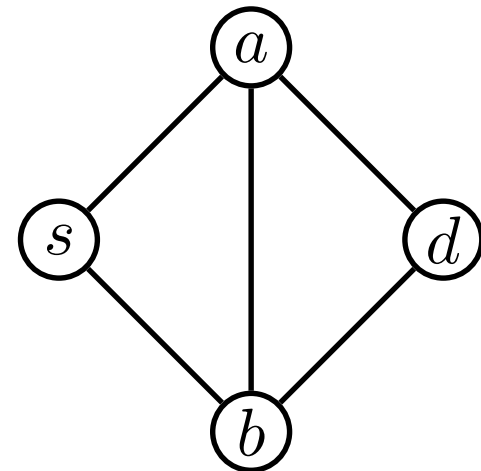
- Ad Hoc On-Demand Distance Vector Protocol
 - routing protocol for wireless mesh networks (wireless networks without wired backbone)
 - Ad hoc (network is not static)
 - On-Demand (routes are established when needed)
 - Distance (metric is hop count)
- developed 1997-2001 by Perkins, Beldig-Royer and Das (University of Cincinnati)
- one of the four protocols standardised by the IETF MANET working group (IEEE 802.11s)

Case Study



- Main Mechanism
 - if route is needed
BROADCAST RREQ
 - if node has information about a destination
UNICAST RREP
 - if unicast fails or link break is detected
GROUPCAST RERR

- performance improvement via
intermediate route reply

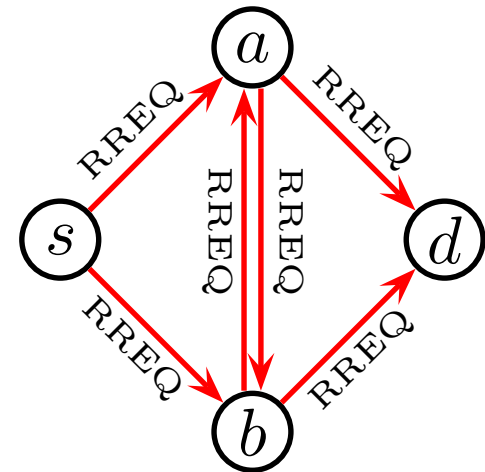


Case Study



- Main Mechanism
 - if route is needed
BROADCAST RREQ
 - if node has information about a destination
UNICAST RREP
 - if unicast fails or link break is detected
GROUPCAST RERR

- performance improvement via
intermediate route reply

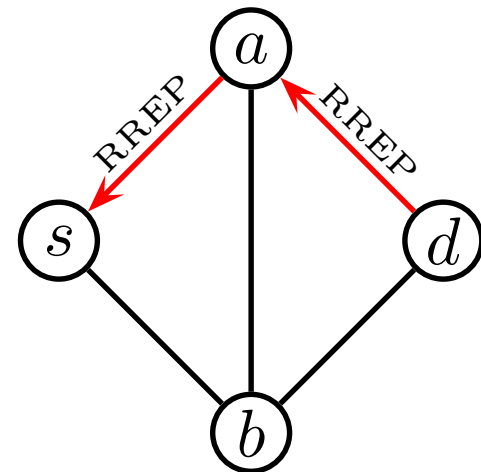


Case Study



- Main Mechanism
 - if route is needed
BROADCAST RREQ
 - if node has information about a destination
UNICAST RREP
 - if unicast fails or link break is detected
GROUPCAST RERR

- performance improvement via
intermediate route reply



Case Study: AODV



```
+ [ (oip, rreqid) ∉ rreqs ]      /* the RREQ is new to this node */
  [[rt := update(rt, (oip, osn, kno, val, hops + 1, sip, ∅))]      /* update the route to oip in rt */
  [[rreqs := rreqs ∪ {(oip, rreqid)}]]      /* update rreqs by adding (oip, rreqid) */
  (
    [ dip = ip ]      /* this node is the destination node */
    [[sn := max(sn, dsq)]      /* update the sqn of ip */
    /* unicast a RREP towards oip of the RREQ */
    unicast(nhop(rt, oip), rrep(0, dip, sn, oip, ip)) . AODV(ip, sn, rt, rreqs, store)
    ▶ /* If the transmission is unsuccessful, a RERR message is generated */
    [[dests := {(rip, inc(sqn(rt, rip))) | rip ∈ vD(rt) ∧ nhop(rt, rip) = nhop(rt, oip)}]]
    [[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)
  + [ dip ≠ ip ]      /* this node is not the destination node */
    (
      [ dip ∈ vD(rt) ∧ dsq ≤ sqn(rt, dip) ∧ sqnf(rt, dip) = kno ]      /* valid route to dip that is fresh enough */
      /* update rt by adding precursors */
      [[rt := addpreRT(rt, dip, {sip})]]
      [[rt := addpreRT(rt, oip, {nhop(rt, dip)})]]
      /* unicast a RREP towards the oip of the RREQ */
      unicast(nhop(rt, oip), rrep(dhops(rt, dip), dip, sqn(rt, dip), oip, ip)) .
```

Case Study: AODV



- full specification of AODV (IETF Standard)
- specification details
 - around 5 types and 30 functions
 - around 120 lines of specification
(in contrast to 40 pages English prose)

Case Study: AODV - Analysis



- *Properties of AODV*

- route correctness



- loop freedom



(at least for some interpretations)

- route discovery



- packet delivery



Case Study: Analysis



- Loop Freedom
 - invariant proof based on about 35 invariants, e.g.

If a route reply is sent by a node ip_c , different from the destination of the route, then the content of ip_c 's routing table must be consistent with the information inside the message.

$$N \xrightarrow{R:*\text{cast}(\text{rrep}(hops_c, dip_c, dsn_c, *, ip_c))} N' \wedge ip_c \neq dip_c \\ \Rightarrow dip_c \in \text{kD}_N^{ip_c} \wedge \text{sqn}_N^{ip_c}(dip_c) = dsn_c \wedge \text{dhops}_N^{ip_c}(dip_c) = hops_c \wedge \text{flag}_N^{ip_c}(dip_c) = \text{val}$$

- ultimately we defined quality on routes the quality strictly increases

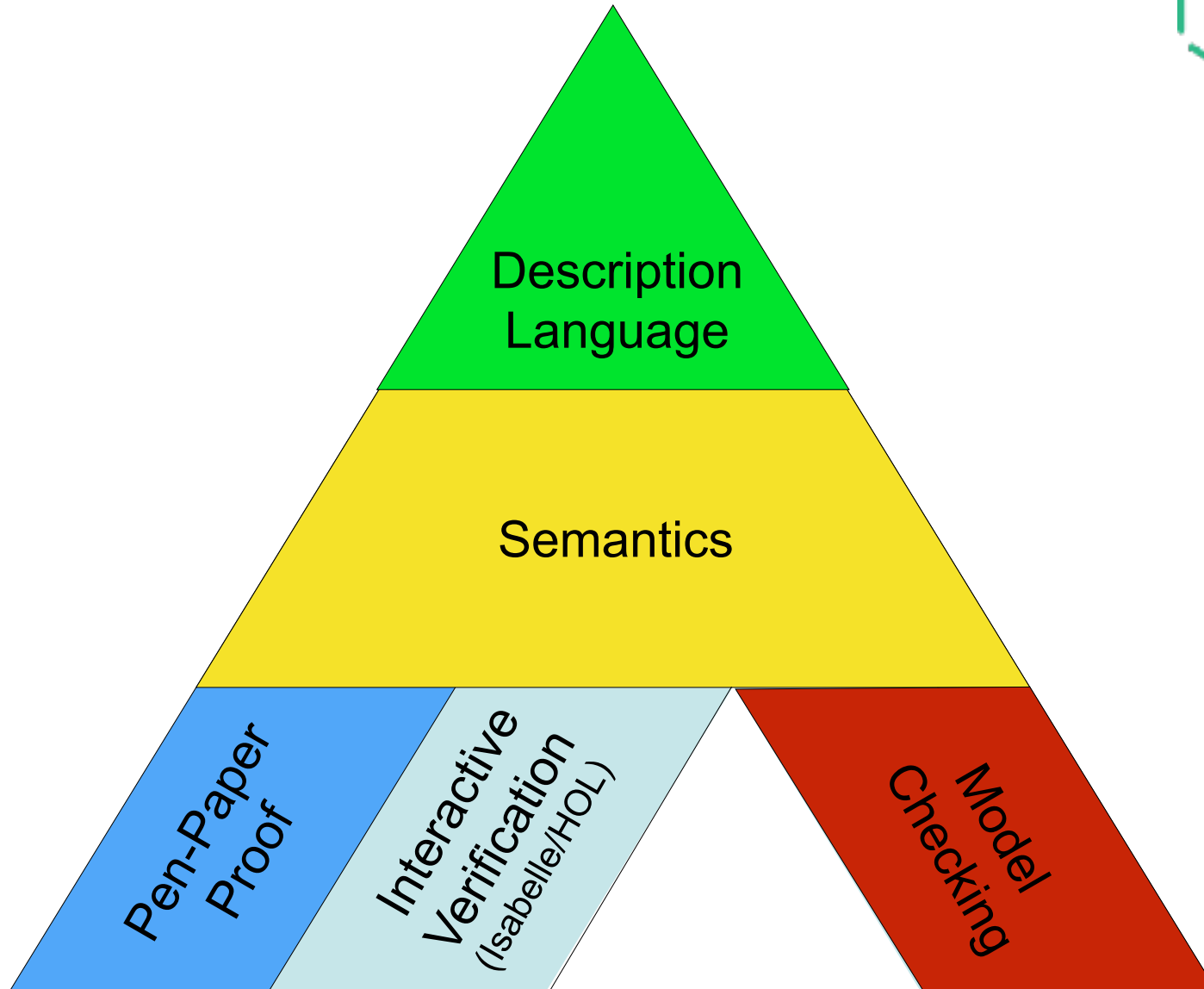
$$dip \in \text{vD}_N^{ip} \cap \text{vD}_N^{nhip} \wedge nhip \neq dip \Rightarrow \xi_N^{ip}(\text{rt}) \sqsubset_{dip} \xi_N^{nhip}(\text{rt})$$

- first rigorous and complete proof of loop freedom of AODV (for some interpretations)

Case Study: Analysis



- Loop Freedom
 - 5184 possible interpretations due to ambiguities
 - 5006 of these readings of the standard contain loops
 - 3 out of 5 open-source implementations contain loops
- Found other shortcomings
 - e.g. non-optimal routing information
 - we proposed solutions and proved them correct

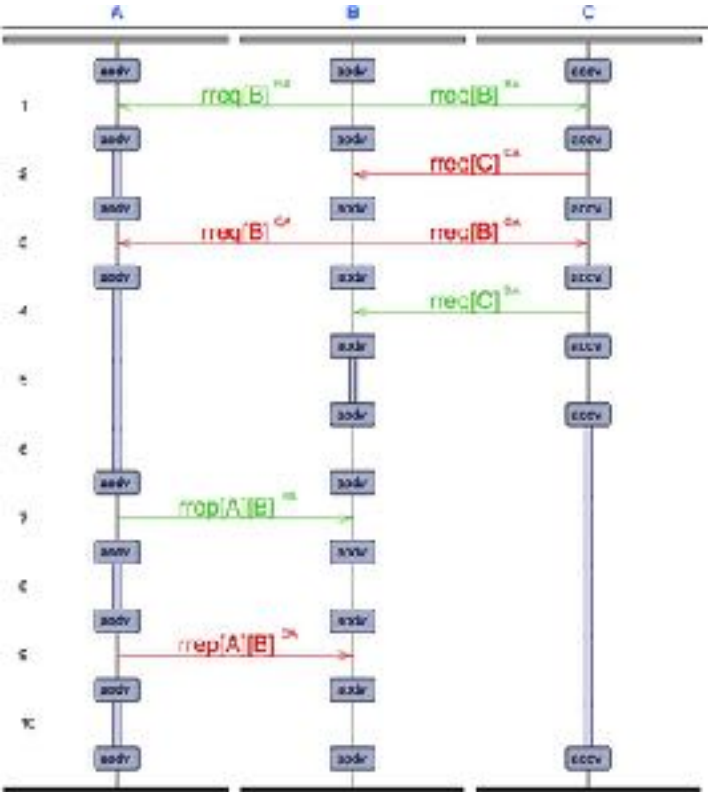
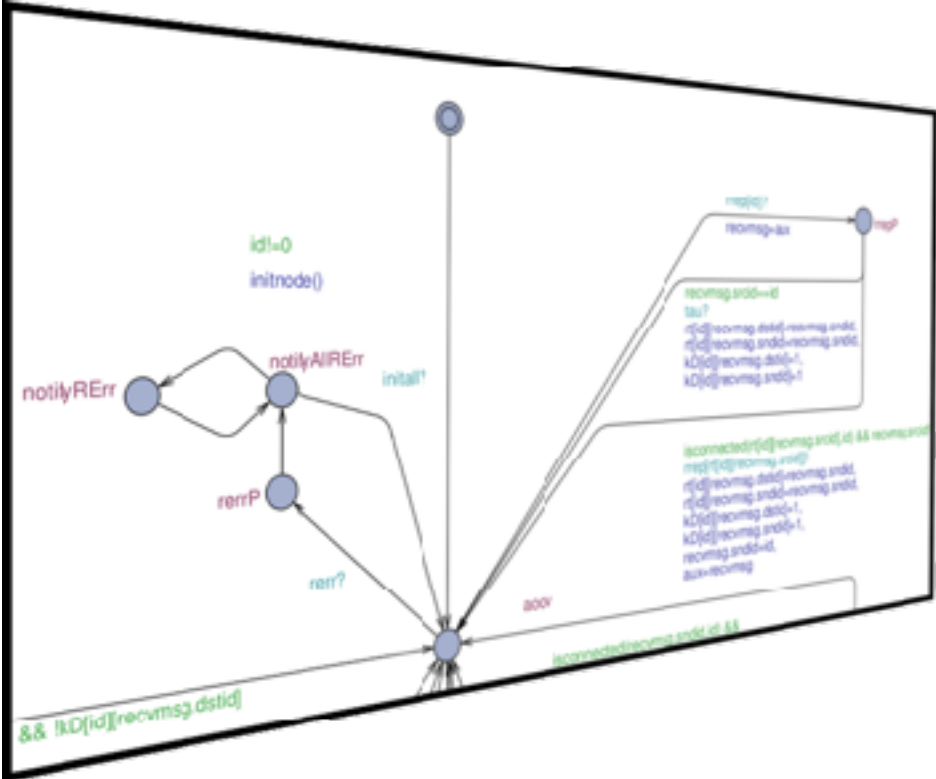


Computer-Aided Verification



- Model Checking
 - quick feedback for development
 - cannot be used for full verification
- (Interactive) Theorem Proving
 - Isabelle/HOL
 - replay proofs
 - proof verification
 - robust against small changes in specification

Model Checking



Model Checking



- Model checking routing algorithms
 - executable models
(generated from process-algebraic specification)
- Complementary to process algebra
 - find bugs and typos in process-algebraic model
 - check properties of specification applied to particular topology
 - easy adaption in case of change
 - automatic verification
- Achievements
 - implemented process algebra specification of AODV
 - found/replayed shortcomings

Isabelle/HOL



```
Isabelle2013-2 - Seq_Invariants.thy (modified)
File Edit Search Windows Folding View Utilities Macros Plugins Help
Seq_Invariants.thy [~/src/HOL/Isabelle/Isabelle2013-2/Seq_Invariants.thy]
220
221 lemma hop_count_positive:
222   'padv i |= on'. Invar (λ(ξ, _) . vizekD (rt ξ) the (dhops (rt ξ) ip) ≥ 1)
223   apply (invar_sims inw add: on_invariant_sims ICF adv_wl advpR_wl (defined))
224   ||
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

prova (prova): step 1

goal (5 subgoals):

```
1. A p [ξ a c] [ξ' pp p']
   l = Padv::B →
   vizekD (rt ξ) . sup it ≤ top (dhops (rt ξ) ip) →
```

Isabelle/HOL



- Generic proof assistant
- We implemented
 - developed process algebra
 - AODV invariant proofs
- Advantages
 - proof verification
 - speed up of analysis of protocol variants
 - analysed variants/improvements more or less automatically
 - quick proof adaption
 - reply of proofs
 - necessary for protocol development

Key Research Outcomes



- New languages and proof methodologies
 - process algebra

- Case Study AODV
 - complete and detailed model (without time)
 - model checking: quick check for counterexamples
 - theorem proving: verification and proof automation



Vision - Practical Protocol Engineering



Design

Verification /
Improvement

```
...are RREQ, i.e. do nothing, ...
update(rt. (sip, 0, val, 1. sip))]. /*update
ip,sn,rt,rreqs,store)
msg = rreq[hops, rreqid, dip, dsn, oip, osn, sip] ^ (oip
/*answer the RREQ with a RREP*/
[rt := update(rt. (oip, osn, val, hops + 1. sip))] /*update
[rreqs := rreqs U {(oip, rreqid)}] /*update the array of
:= max(sn, dsn)] /*update the sqn of ip
[rt := update(rt. (sip, 0, val, 1. sip))] /*update the route
unicast(zhcp(rt.oip),rrep(0,dip,sn,oip,ip))
ADDV(ip,sn,rt,rreqs,store)
+ [msg = rreq[hops, rreqid, dip, dsn, oip, osn, sip] ^ (oip, rreqid
(dip ∈ vB(rt) ∨ sqn(rt,dip) < dsn ∨ sqnf(rt,dip) = unk) ]
/*forward RREQ*/
[rt := update(rt. (oip, osn, val, hops + 1. sip))] /*update
[rreqs := rreqs U {(oip, rreqid)}] /*update the array
:= update(rt. (sip, 0, val, 1. sip))] /*update the
dcast(rreq[hops + 1,rreqid,dip,max(sqn(rt,d
r,rreq,store)
rreqid,dip,dsn,oip,osn,sip) ^
```

Implementation

Future Work



- Research (1)
 - timed analysis
 - build tool suite
 - better tool support (more proof automation)

- Research (2)
 - code generation
 - code verification

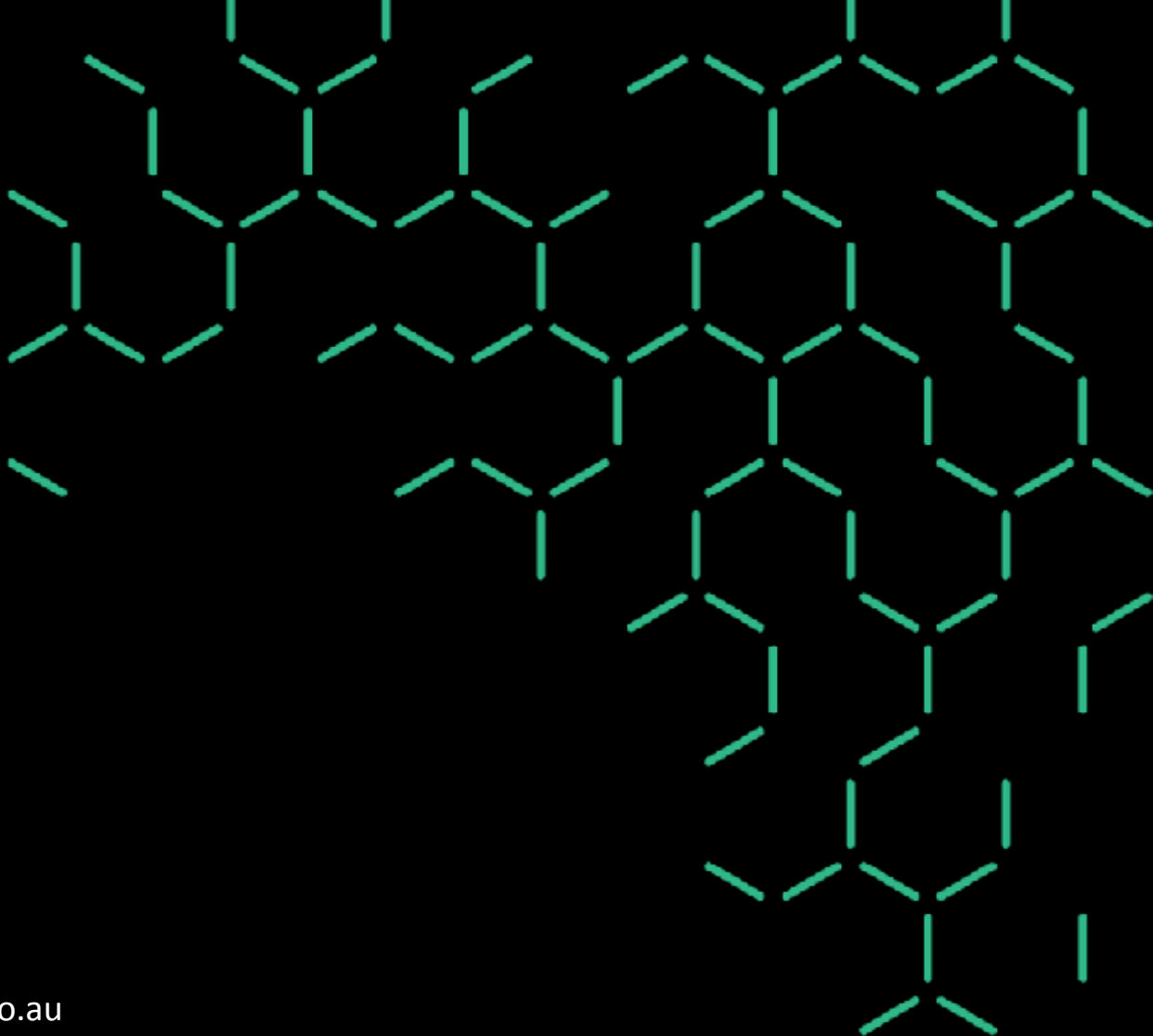
- Training
 - train network engineers to use our approach
 - hardest to achieve

Questions?



“Despite the maturity of formal description languages and formal methods for analyzing them, the description of real protocols is still overwhelmingly informal. The consequences of informal protocol description drag down industrial productivity and impede research progress”.

Pamela Zave (AT&T)



Trustworthy Systems
Rob van Glabbeek

t +61 2 9490 5938
e Robert.vanGlabbeek@data61.csiro.au
w www.data61.csiro.au

www.data61.csiro.au

