

Application of Coloured Petri Nets to Protocols



Tutorial presented at
Fifth Workshop and Tutorial on Practical Use of
Coloured Petri Nets and CPN Tools

Copyright 2004 by
Jonathan Billington
Computer Systems Engineering Centre
University of South Australia, Adelaide, Australia

Lars M. Kristensen
University of Aarhus, Denmark

9 October 2004



AGENDA

9.00 Introduction (JB)

- Introduction to Protocol Engineering
- Protocol Verification Methodology

10.30 Break

11.00 Application to Internet Protocols (JB)

- TCP
- IOTP

12.30 Lunch

14.00 Mobile Ad-hoc Networks (LMK)

15.30 Break

16.00 Industrial Perspectives (Ericsson, JK)

16.30 Close

INTRODUCTION TO PROTOCOL ENGINEERING

- What are Protocols?
- Examples
- Protocol Architectures
- Protocol Layering Terminology
- Protocol Engineering
- Computer Aided Tools for PE
- Literature
- Summary

WHAT IS A PROTOCOL ?

- Taken from Diplomacy

Customs and regulations dealing with the ceremonies and etiquette of the diplomatic corps and others at a court or capital.

Macquarie Dictionary

- Network Protocols

- Rules which govern the way parties interact in distributed Systems
- First coined in April 1967 by Scantlebury and Bartlett at NPL, UK in a memo

Long History of Signalling

- Dates back over 2500 years
- Use of fire signals
 - Communicated fall of Troy to Athens
 - About 300 miles
- Polybius torch code
 - 2nd century BC
 - Transmitting Greek alphabet
 - 2 sets of 5 torches, plus a screen

Lesson

It is chiefly unexpected occurrences which require instant consideration and help

Polybius, 2nd century B.C.

Expect the unexpected !

Gerald Holzmann, 1991

Protocol Definition: Elements

- Purpose/Requirements (Service Specification)
- Communication medium characterization
- Message set
- Encoding (syntax/format) for each message
- Communication Procedures

EXAMPLES

- Internet/IETF (PPP, IP, TCP, UDP, DCCP, SCTP, HTTP, FTP, SMTP, DNS etc)
- QoS (RSVP, RTP)
- E-Commerce (eg IOTP)
- Multimedia over packet (H.323, H.245, RTP, SIP)
- Wireless (Wireless Application Protocol, Bluetooth, WiFi etc)
- Integrated Services Digital Network (ISDN) (Q.900 series)
- B-ISDN (ATM) - multimedia, multiparty
- Telephony Line Signalling, Common Channel Signalling (Q.700 series)
- Network Management (TMN, SNMP)
- LANS/MANS (IEEE 802.x)
- Middleware/Open Distributed Processing (eg CORBA, SOAP, Traders)
- Grid Computing
- Web services (W3C)

Protocols increasing in Number and Complexity



Protocol Architectures

- Protocols are normally designed in layers
 - Manage complexity
 - e.g. TCP/IP, OSI, CCSS 7, WAP

TCP/IP Reference Model



Application

Transport

Network

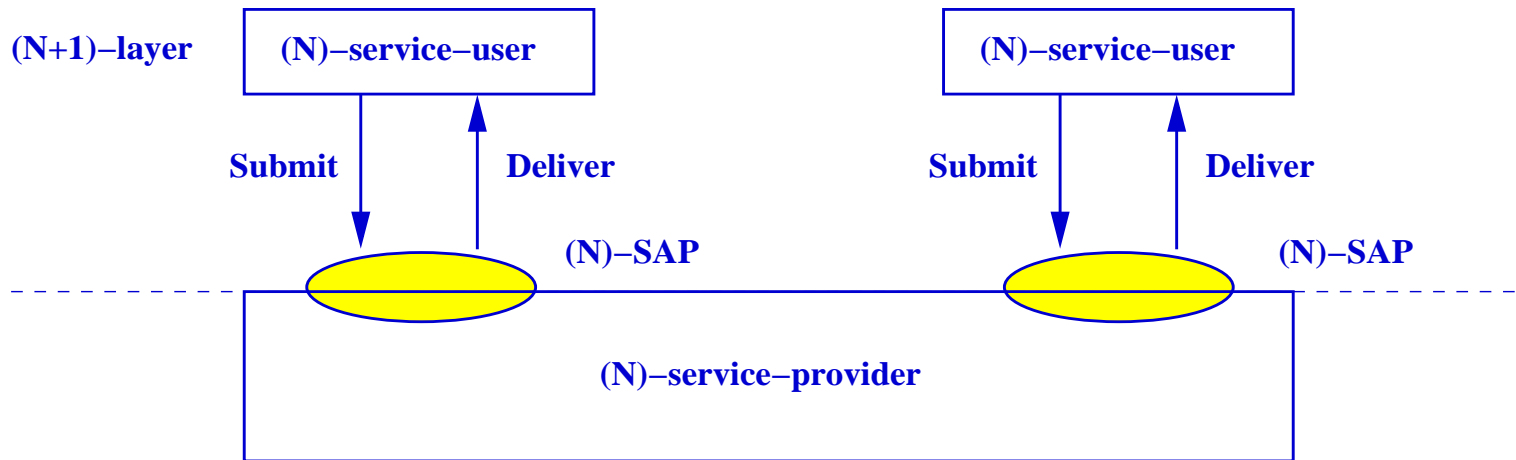
Data link

Physical

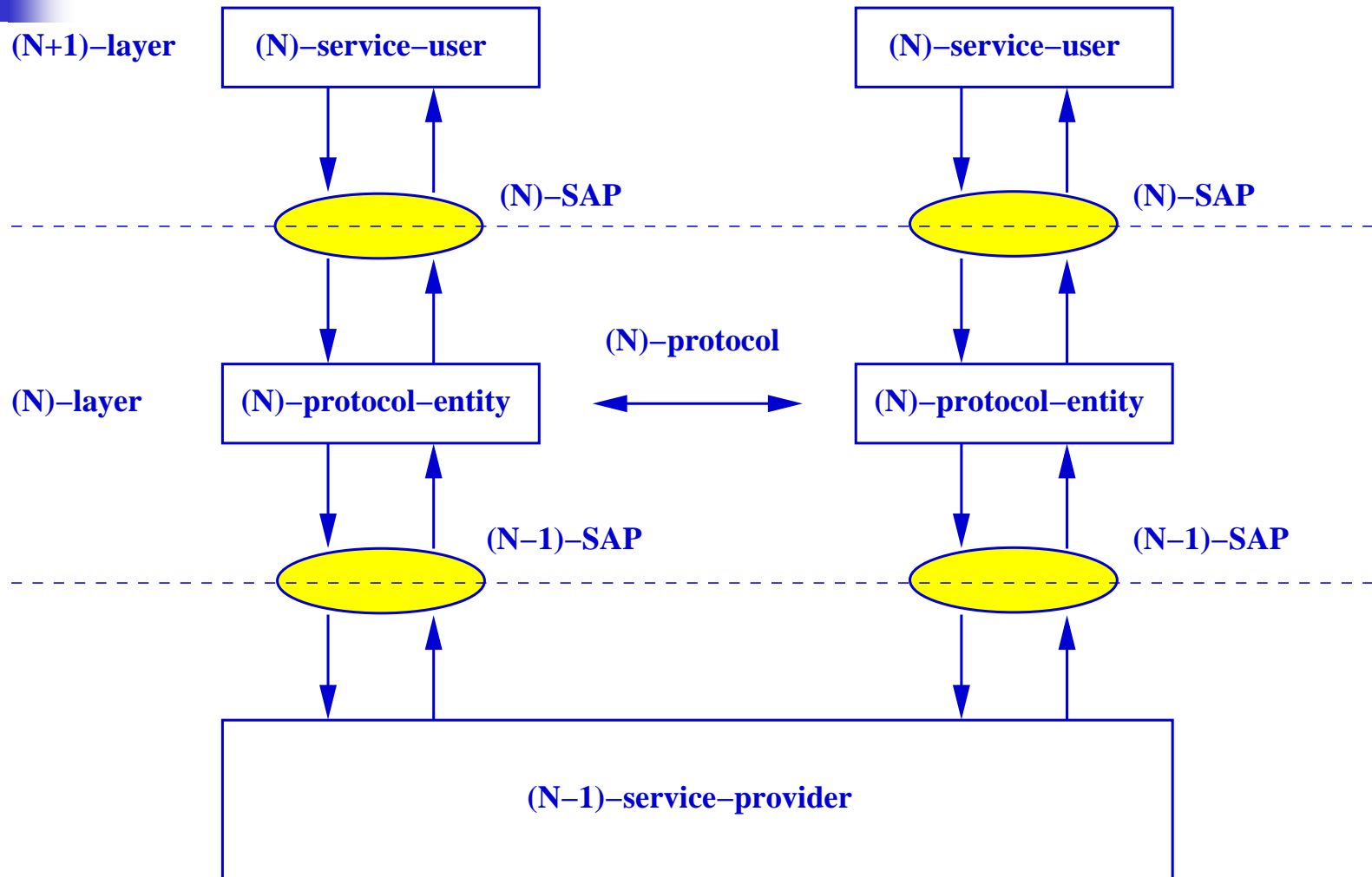
Open Systems Interconnection (OSI)

- Developed generic layering terminology
 - Services, service user and provider, service data units
 - Protocol, protocol entity, protocol data units
 - 'Onion model'

OSI Service Model



OSI Protocol Model





Problems with Protocols

- Ambiguous specifications
- Little analysis before implementation
- Lack of design rules
- Lack of integration of real-time and functional behaviour
- Lack of rigorous development methodologies
- Lack of integrated computer tool sets
- Lack of expertise

PROTOCOL ENGINEERING

- A **Discipline** for the rigorous design, implementation and maintenance of communication protocols throughout life
- Part of (Distributed) Systems Engineering
 - Concurrency theory (discrete mathematics)
 - Computer Science (algorithms, CAD)
 - Software Engineering methodologies
 - Performance Evaluation (e.g. stochastic processes)
 - Protocols
- A new discipline? (IEEE Software, Network Protocols, Jan 92)
 - No - term coined in 1981 by Piatkowski
 - Yes - only recently starting to be taught

DEFINITIONS OF PE ACTIVITIES

- Protocol Synthesis:
 - Formally deriving a protocol specification from the service required and the underlying service.
 - Protocol design rules that ensure correct operation.
- Protocol Verification:
 - The proof that the protocol operating over the underlying service does provide the required service.
- Performance Analysis:
 - Analyse protocol specifications to determine throughputs, delays, reliability and availability
 - May be used to determine buffer sizes and bottlenecks

DEFINITIONS (cont.)

- Automatic Implementation:
 - The process of transforming a specification into an implementation automatically.
- Conformance Testing:
 - Testing implementations to see if they conform with their protocol specifications.
- Protocol Maintenance:
 - Upgrading or replacing protocols or parts of a protocol architecture
- Protocol Conversion:
 - Translation between two different protocol architectures

IMPORTANCE OF PE -1

- Vast area - vital to realisation of distributed systems and advanced information services
- Improves the design and testing of protocols
 - unambiguous, analysed specifications
 - more efficient and thorough testing
 - cost savings (less testing / debugging / maintenance)
 - improved customer satisfaction
 - reduce (eliminate?) hand-coding errors
 - minimize costly outages
 - decrease the chance of litigation

IMPORTANCE OF PE - 2

- Protocol Education via CAD Tools
- Wide Range of Applications
 - Telecommunication signalling and services
 - Service interaction problems
 - Distributed Processing
 - Multiprocessor systems
 - Fault Tolerance
 - Concurrent Programming
- Contribution to International Standards
 - Methodologies/Techniques
 - Improved Protocol Specifications

Requirements of Formal Techniques

- Well-defined
- Sufficiently expressive
- Analysable
- Support refinement
- Implementation independence
- Facilities for Abstraction
- Complexity management
- Support protocol life cycle, including rapid prototyping
- Support automation
- Strongly related to physical system for ease of model validation

Net Techniques for Specification

- Hierarchical nets for structure and behaviour, e.g.
 - Hierarchical CP-nets
- High-level Nets for Dynamic Behaviour
 - CP-nets or PrT nets or Algebraic Nets
 - P/T-nets for simple examples
- Stochastic and Timed PNs for performance evaluation, e.g.
 - GSPNs, DSPNs

Computer Aids

- Editors (graphical and textual)
- Specification, Animation and Simulation
- Analysers, debuggers and verifiers
- Performance Analysers
- Translators
 - Synthesiser
 - Code generator
- Implementation Testers

LITERATURE

- G.J. Holzmann, Design and Validation of Computer Protocols, Prentice Hall, 1991.
- J. Billington, M. Diaz, G. Rozenberg (Eds), Application of Petri Nets to Communication Networks, LNCS 1605, 1999.
- K. Jensen, Coloured Petri Nets, 3 Volumes, Springer, 1997.
- Int. Conferences on Application and Theory of Petri Nets, published in LNCS
- Advances in Petri Nets, LNCS
- FORTE - Int. Conf. on Formal Techniques for Networked and Distributed Systems
- IFIP Int. Workshop on Protocol Test Systems
- Int. Conference on Computer aided Verification (CAV)
- CONCUR
- TACAS
- IFIP Int. Symposium on Protocol Specification, Testing and Verification (now FORTE)



Summary

- Overview of the field of PE
- Protocols are a vast and growing domain
- Importance of Formal Techniques
- A look at PE computer tools

Rest of the tutorial concentrates on

- Protocol Specification and Verification using high-level nets



HLPN based Protocol Verification Methodology



Outline

- Overview of Methodology
- High-level Nets
- Occurrence Graphs
- Finite State Automata
- Conclusions

Introduction to Protocol Verification

- A protocol needs to satisfy the requirements stated in its service specification, e.g.
 - a reliable data transfer protocol delivers data in the order it was sent, without loss, duplication, corruption or misdelivery (safety) and in a timely manner (progress - i.e. eventually, all data sent is received)
 - a connection establishment protocol successfully establishes a connection (progress) and can not enter any bad states (safety)
- Service specifications have been written for the Open Systems Interconnection protocols in a rigorous way – e.g. the OSI Transport Service
- Service specifications are written according to conventions standardized in ITU-T Recommendation X.210

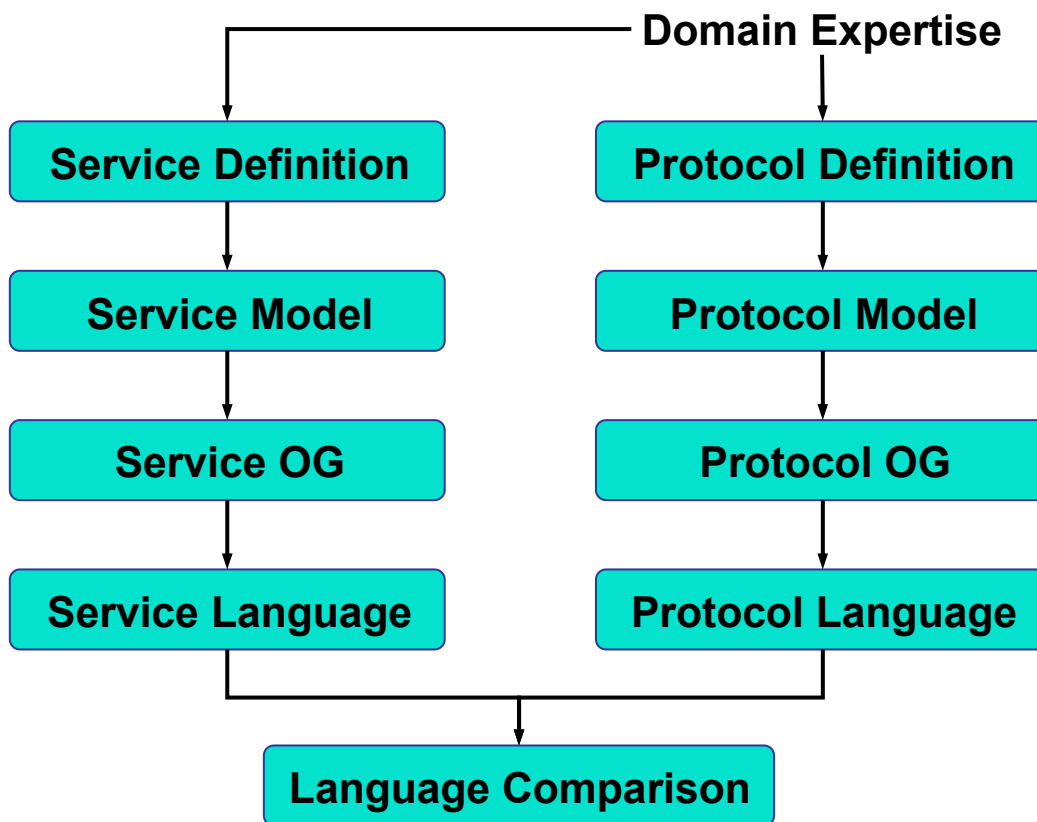
Service Specifications

- Many protocols are defined without a service specification – particularly Internet Protocols (bottom-up approach)
 - “Rough consensus and running code”
- Service specifications attempt to encapsulate the users’ view of what is wanted from the protocol
- Service Model
 - Service Users
 - Service Provider
 - Service primitive – abstract representation of the interactions between a user and provider
 - Groups of service primitives for a service feature, such as connection establishment or data transfer

Service Specifications

- OSI Service specifications define
 - a group of service features
 - a set of service primitives
 - associated data as parameters of the primitive
 - sequencing of primitives at the local interface (sequence tables or state transition diagrams or tables)
 - end to end (global) sequencing via time sequence diagrams – invariably incomplete
- First step towards verification
 - Prove that the protocol does obey the sequencing constraints defined in the service
 - How?

Protocol Verification Methodology



Methodology: Defns and Models

- Create Service and Protocol Definitions
 - Protocol obtained from a standards document (ISO/IEC, ITU-T, IETF, WAP Forum, OMG etc)
 - Service may be defined in a standard (ISO/IEC, ITU-T or WAP) but may not be (IETF)
 - Underlying service, or “medium” over which the protocol operates, also needs to be defined
- Build two HLnet models
 - Service to be provided by the protocol
 - Protocol operating over the underlying service
- Associate service primitives with transitions in both the service and protocol

Methodology: OGs

- Generate the Occurrence Graph (OG) (State Space) for both the Service and Protocol HLnet models
 - If finite, use tools like Design/CPN or CPN Tools
 - Limitation – parameters need to be fixed
 - If not bounded ?
 - Possible to limit parameters, such as sequence numbers, retransmission counters, window sizes and channel capacities to small values
 - Use parametric/infinite state verification tools (FAST, TREX)
 - Often need to use advanced RA techniques
 - Equivalence classes/Symmetry
 - Sweepline
 - May be able to use recursive techniques and hand proofs to generalise

Methodology: FSAs

- Transform the OG to an FSA
 - for the service and protocol
 - Designate initial state and final states
 - Initial state = Initial marking of HLnet
 - Final states – (depends on protocol)
 - Often dead markings (this is when the system stops)
 - Could be initial marking
 - Replace transitions not labelled by primitives by internal or null transitions
 - Map these transitions to epsilon (ϵ) transitions

Methodology: Deterministic FSA

- Transform the FSA derived from the OG to its equivalent (minimum) deterministic FSA
 - For both service and protocol
 - Using standard automata reduction techniques
 - 5 steps (see e.g. Barrett and Couch)
 - Remove empty transition cycles
 - Remove empty transitions
 - Determinise
 - Remove inaccessible states
 - Minimise
 - Use a tool such as FSM from ATT

Methodology

Service and Protocol Languages

- Determinised FSAs
 - encapsulate the set of sequences of primitive occurrences
 - for both the service and the protocol
 - hence define
 - **Service language (SL)**: the set of sequences of primitive occurrences that occur in the **service**
 - **Protocol language (PL)**: the set of sequences of primitive occurrences that occur in the **protocol**

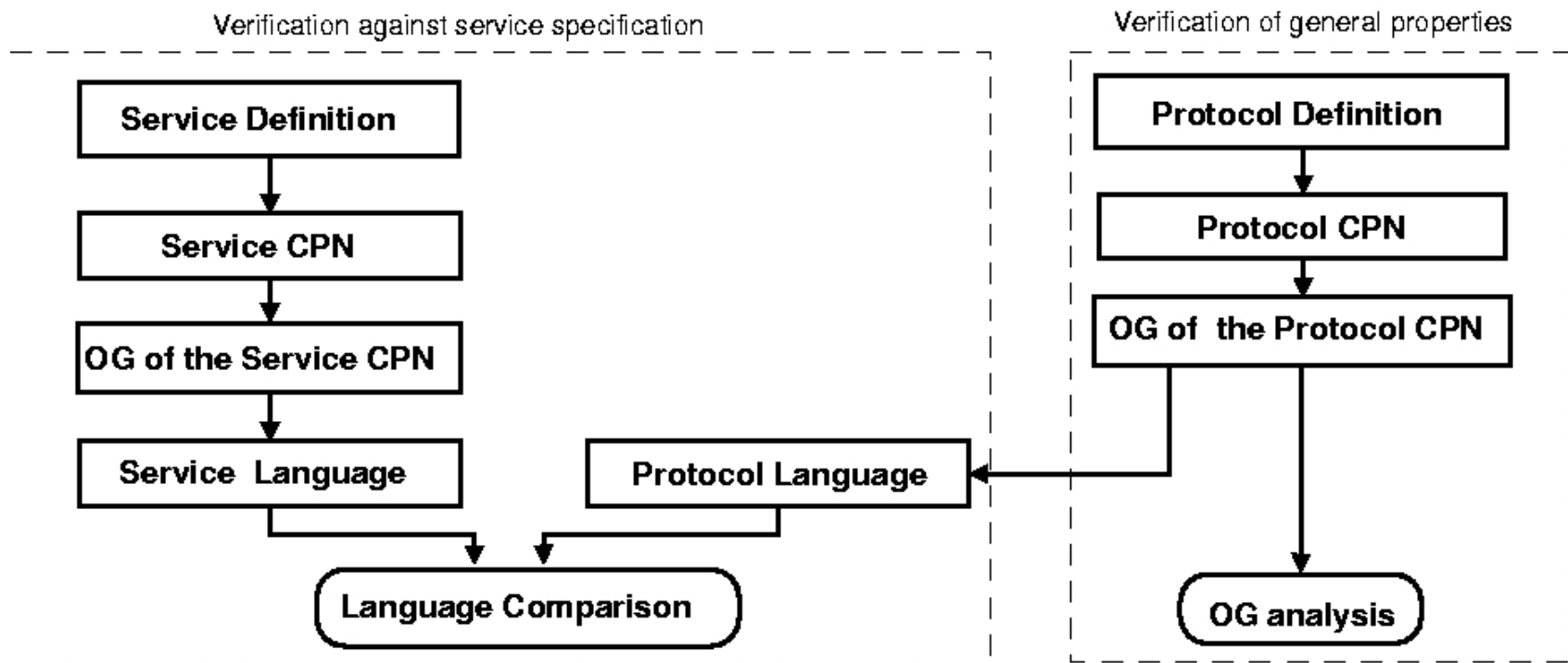
Methodology: Language Comparison

- Deterministic FSAs can be compared for
 - Language equivalence; and
 - Language inclusion
- Compare Service and Protocol Languages
 - If equivalent, then the protocol satisfies the sequencing constraints of the service
 - If PL subset of SL, then this may be acceptable
 - e.g. in the case of options or interleaved events
 - Need to define minimum acceptable subsets
 - If PL is **not** a subset of SL
 - Sequences in the protocol not defined in the service
 - Hence **error**, either in service or in the protocol
 - Use a tool like FSM to do this comparison

Other Properties

- Generating the OG and Automata reduction allow us to determine sequencing compatibility
- What about general properties?
 - Deadlocks
 - Livelocks
 - Forbidden states
 - Bounded buffers
- Other protocol specific properties?
- Approach:
 - Define the properties; and
 - Query the OG (Model Checking)

PV Methodology Extension



Deadlocks

- The leaf nodes of the OG correspond to dead markings
- These are terminal states of the protocol
- For some protocols, termination is required
 - e.g. Connection Management Protocols
- Notion of:
 - Desired terminal state
 - Undesired terminal state (**Deadlock**)
- Tools such as Design/CPN have built in queries for
 - Dead markings
 - Home spaces

Livelocks

Cyclic behaviour that does not make progress

- Detect by generating the Strongly Connected Components (SCC) of the OG
- Terminal SCCs:
 - Dead markings or
 - Represent cyclic behaviour
 - Desired, e.g. main loop
 - Undesired (**livelock**)
- Sufficient condition: Net is livelock free if all terminal SCCs are dead markings
 - All dead markings form a home space
- Calculation of SCCs is fast, and provided by tools such as Design/CPN (can be done on the fly)

High-level Nets

- Standard: ISO/IEC 15909
 - Part 1: Concepts, Definitions and Graphical Notation (FDIS)
 - CPN semantics
 - Graphical form (uses signatures)
 - Mapping from HLPN Graph to CPN semantics
 - Part 2: Transfer Format (PNML)
 - XML based
 - Now a WD, with CD planned for mid 2005
 - Defacto Editor is Ekkart Kindler
 - Part 3: Extensions (Future)
- Base definition on standard

High-level Net Definition

$HLPN = (P, T, D; \text{Type}, \text{Pre}, \text{Post}, M_0)$

- P is a finite set of Places
- T is a finite set Transitions disjoint from P
- D is a non-empty finite set of non-empty domains (sets) where each element of D is called a type
- $\text{Type}: P \rightarrow D$ is a function used to assign types to places and to determine transition modes
- $\text{Pre}, \text{Post}: T \rightarrow \mu\text{PLACE}$ are the pre and post mappings
 - $\text{TM} = \{(t, m) \mid t \in T, m \in \text{Type}(t)\}$ (transition modes)
 - $\text{PLACE} = \{(p, g) \mid p \in P, g \in \text{Type}(p)\}$ ('unfolded places')
 - μPLACE is the set of multisets over the set, PLACE
- $M_0 \in \mu\text{PLACE}$ is the initial marking of the net

Occurrence Graphs

An occurrence graph of a HLPN is a labelled directed graph

- $OG = (V, TM, A)$ where
 - $V = [M_0 >$ is the set of markings reachable from M_0 (the reachability set);
 - TM is the set of transition modes of the HLPN; and
 - $A = \{(M, tm, M') \in V \times TM \times V \mid M[tm > M'\}$ is a set of arcs labelled with transition modes.

Abstract OGs

- May only be interested in transition names
- not the mode (bindings of the variables)
 - e.g. in the case of service primitives, where we have abstracted from parameter values
- Then an *abstract* OG of a HLPN, with respect to transitions, is a labelled directed graph:
- $AOG^T = (V, T, A)$ where
 - $V = [M_0 >$
 - T is the set of transitions of the HLPN; and
 - $A = \{(M, t, M') \in V \times T \times V \mid (t, m) \in TM \text{ and } M[(t, m) > M']\}$ is a set of arcs labelled by transitions.

Abstract OGs

- Only interested in the identification of the markings
- Introduce an injection, mapping reachable markings into the set of positive integers:
- $I: [M_0 > \longrightarrow \mathbb{N}^+$ where
 - $I(M_0) = 1$ represents the initial marking.
- Then an *abstract* OG of a HLPN with respect to markings, is a labelled directed graph:
- $AOG^M = (V, TM, A)$ where
 - $V = \{I(M) \mid M \in [M_0 >\}$ is the set of nodes
 - TM is the set of transition modes of the HLPN; and
 - $A = \{(I(M), tm, I(M')) \in V \times TM \times V \mid M[tm > M'\}$ is a set of arcs labelled by transition modes

Abstract OGs

Combine both abstractions

- An *abstract* OG of a HLPN with respect to transitions and markings, for Injective Mapping I , is a labelled directed graph:
- $\text{AOG}^{\text{TM}} = (V, T, A)$ where
 - $V = \{I(M) \mid M \in [M_0 >]\}$ is the set of nodes
 - T is the set of transitions of the HLPN; and
 - $A = \{(I(M), t, I(M')) \in V \times T \times V \mid (t, m) \in \text{TM} \text{ and } M[(t, m) > M']\}$ is a set of arcs labelled by transitions

Finite State Automata

- Mapping an (abstract) OG to its FSA
- FSAs used for language comparison
- Not concerned with details of markings
- Hence use an abstract OG, just represent the nodes by positive integers
- Often label transitions in HLPN with service primitive names, however ...
- Convenient to have a general mapping from transition modes to service primitives

Mapping Modes to Primitives

- Need a function that maps each transition mode in the abstract OG to
 - a service primitive name, or
 - an epsilon ϵ
- Prim: $TM' \longrightarrow SP \cup \{\epsilon\}$
 - TM' is the subset of transition modes used to label arcs in the abstract OG; and
 - SP is the set of service primitive names

OG to FSA Transformation

Given $\text{AOG}^M = (V, \text{TM}, A)$ with injection, I , and primitive mapping, Prim , the corresponding FSA is

$\text{FSA}_{\text{AOG}^M} = (V, \text{SP} \cup \{\varepsilon\}, A_{\text{SP}}, v_0, F)$ where

- V is the nodes of AOG^M (the states of the FSA);
- SP is the set of service primitive names of the system of interest (the alphabet of the FSA);
- $A_{\text{SP}} = \{(v, \text{Prim}(tm), v') \mid (v, tm, v') \in A\}$ is the set of transitions labelled by service primitives or internal events (the transition relation of the FSA);
- $v_0 = 1$ corresponds to the abstract initial marking (initial state of the FSA); and
- F , a subset of V , is the set of final states.

Conclusions

- Introduced the area of Protocols
 - Large and growing area
 - Life blood of modern economies
- Protocol Engineering terminology
- Need for rigorous methods
- Supported by tools and methodologies
- Introduced a verification methodology
 - Based on high-level nets and automata
 - Given basic definitions
 - Billington et al, 'A Coloured Petri Net Approach to Protocol Verification', LNCS 3098, pp 210-290, 2004