Networking and Communications

Grid Computing (M)

Lecture 1



Lecture Outline

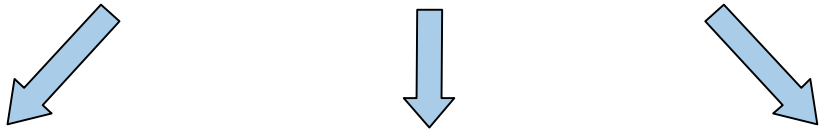
- What is grid computing?
- Module structure and administration
  - Aims and objectives
  - Intended learning outcomes
  - Prerequisites
  - Timetable
  - Lecturers
  - Reading list
  - Assessment
- Networking and communications
  - Layered protocol architectures
  - Review of IP networks and protocols
  - The Berkeley sockets interface
  - Higher layer protocols

Copyright © 2006 University of Glasgow

What is Grid Computing?

"A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities" enabling "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations"

Infrastructure for Internet-scale Distributed Systems



Authentication, authorisation and accounting

Scalability and heterogeneity

Large-scale and multi-organisation resource management

Copyright © 2006 University of Glasgow

Course Aims and Objectives

- To provide the participants with:
  - Detailed understanding of the key problems and issues that arise when attempting large-scale distributed computation, within organisations & across organisational boundaries
  - Insight into the architectural implications of Grid-scale computation
  - Awareness of current research issues in:
    - Grid architecture and infrastructure
    - Scalable distributed computation
    - Integration of applications across autonomous organisations
  - Practical experience of current Grid technologies and associated standards
  - Skills in utilising current Grid tools and technologies
  - Appreciation of the weaknesses of existing tools and technologies, and potential areas for improvement

Copyright © 2006 University of Glasgow

Intended Learning Outcomes

- By the end of this module, participants should be able to:
  - Critically discuss and reason about large-scale distributed system architectures, infrastructures and technologies
  - Articulate research challenges in multi-organisational distributed computing, including Grid computing
  - Design and implement Grid computing applications using Globus or similar toolkits
  - Justify the applicability, or non-applicability, of Grid technologies for a specific application

Copyright © 2006 University of Glasgow

Prerequisites

- Lectures will assume as background degree level knowledge of:
  - Operating systems
  - Distributed algorithms and systems
  - Communications and networks
  - Databases and Internet technologies

OS3, NSA3, DAS4, NCT4 and DBIT4

The introductory lectures will rapidly revise these to aid those who have incomplete background

- Practical work requires use of Java and C programming in a Unix environment, and assumes familiarity with engineering principles underpinning non-trivial system construction

Copyright © 2006 University of Glasgow

Prerequisites and Focus

- Students are expected to learn quickly, and to master complex systems, languages and technologies in a self-directed manner
- Focus *will not* be on teaching Grid Computing languages and technologies as such
  - You should be competent programmers who can take software and trial it out yourself; "some" training given on technologies, languages, etc.

- Focus *is* on understanding the fundamental computing science topics underlying Grid Computing
  - Why? Grid Computing is a highly dynamic area, where the standards, technologies, and software change all of the time
  - You should understand Grid Computing concepts, and be able to apply them to various scenarios, using a mix of technologies

Copyright © 2006 University of Glasgow

Timetable

Week starting:	Tue 12:00-13:00	Wed 12:00-13:00	Fri 11:00-12:00
9 Jan	Lecture 1	Lecture 2	Lecture 3
16 Jan	Lecture 4	Lecture 5	Lecture 6
23 Jan	Lecture 7	Lecture 8	Tutorial 1
30 Jan	Lecture 9	Lecture 10	Tutorial 2
6 Feb	Lecture 11	Lecture 12	Tutorial 3
13 Feb	Lecture 13	Lecture 14	Tutorial 4
20 Feb	Tutorial 5	Tutorial 6	Tutorial 7
27 Feb	Lecture 15	Lecture 16	Tutorial 8
6 Mar	Lecture 17	Lecture 18	Tutorial 9
13 Mar	Lecture 19	Lecture 20	Tutorial 10

Copyright © 2006 University of Glasgow

Timetable

Week 1	Lecture 1	Networking and Communications	(csp)
	Lecture 2	Remote Procedure Calls	(pd)
	Lecture 3	Distributed Systems	(pd)
Week 2	Lecture 4	Systems Architecture	(pd)
	Lecture 5	Mark up Languages and XML	(femi)
	Lecture 6	Web services	(femi)
Week 3	Lecture 7	Large Scale Systems Architecture	(csp)
	Lecture 8		
	Tutorial 1	Introduction to Globus	(ros)
Week 4	Lecture 9	Security	(ros)
	Lecture 10		
	Tutorial 2	Basic client/server development with Globus	(ros)
Week 5	Lecture 11	Resource Management	(pd)
	Lecture 12		
	Tutorial 3	Introduction to Globus security infrastructure	(ros)

Copyright © 2006 University of Glasgow

Timetable

Week 6	Lecture 13	Scheduling and Accounting	(csp)
	Lecture 14		
	Tutorial 4	Introduction to Condor	(ros)
Week 7	Tutorial 5	Introduction to Permis and SAML	(ros)
	Tutorial 6	Introduction to OGSA-DAI	(ros)
	Tutorial 7	Q&A on programming assignment	(ros)
Week 8	Lecture 15	Scalability and Heterogeneity	(joe)
	Lecture 16		
	Tutorial 8	Introduction to portal technologies	(ros)
Week 9	Lecture 17	Systems Modelling and Simulation	(joe)
	Lecture 18		
	Tutorial 9	Review of example systems	(ros)
Week 10	Lecture 19	Summary and Review	(csp)
	Lecture 20	Future Directions	(csp)
	Tutorial 10	Future Directions	(ros)

Copyright © 2006 University of Glasgow

Lecturers



Dr. Colin Perkins – module coordinator
csp@dcs.gla.ac.uk



Dr. Peter Dickman
pd@dcs.gla.ac.uk



Prof. Joe Sventek
joe@dcs.gla.ac.uk



Prof. Richard Sinnott
ros@dcs.gla.ac.uk



Dr. Olufemi Komolafe
femi@dcs.gla.ac.uk

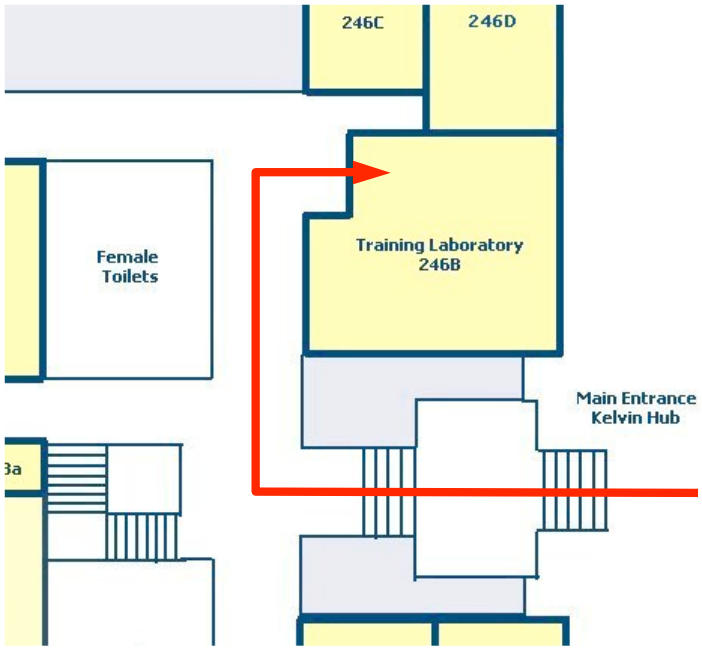
Module taught by a large team to cover the broad range of the subject.

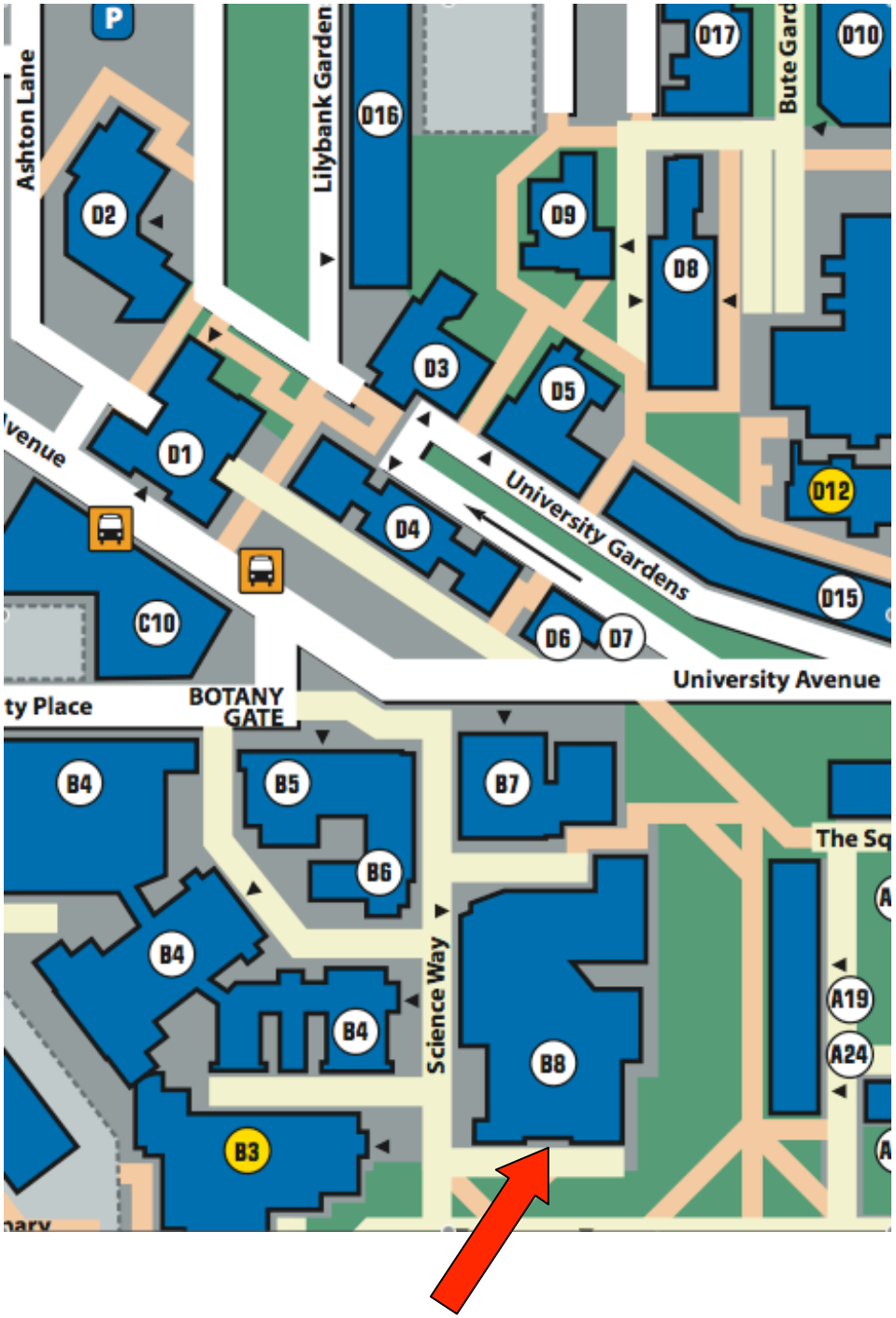
Willing to discuss the module and answer questions – *make appointments by email.*

Copyright © 2006 University of Glasgow

Tutorials

Note: tutorials and laboratory sessions will take place in room 246B in the Kelvin Building (and *not* in Boyd Orr 513 as originally timetabled)





Copyright © 2006 University of Glasgow

Reading List

- There is no set text for this module. Research papers will be distributed as required, and technical manuals and related documentation will be issued as part of the practical activities.

http://csperkins.org/teaching/grid/

Copyright © 2006 University of Glasgow

Assessment

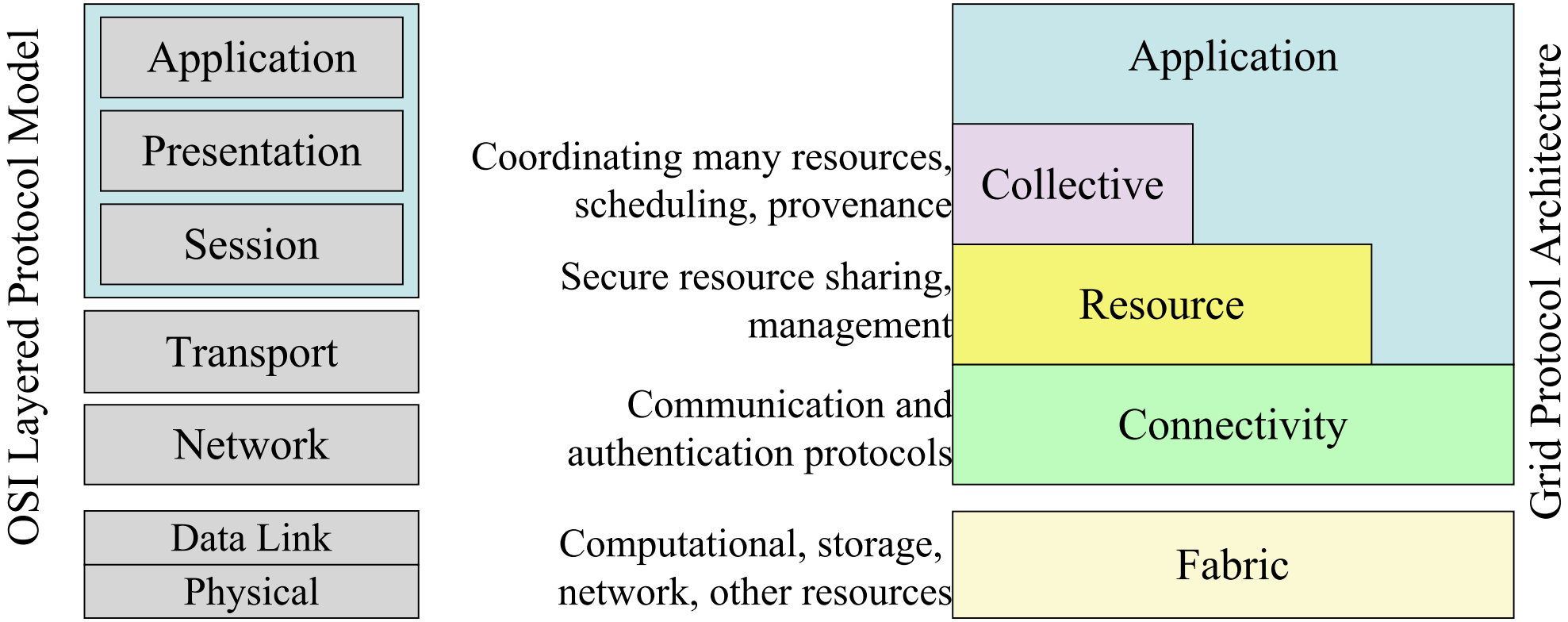
- Level M module, worth 10 credits
- 70% Examination
  - Answer 1 mandatory wide ranging question
  - Answer 2 out of 4 optional and more narrowly focussed questions
- 30% Coursework
  - 3 small exercises, each marked 0 or 1:
    - Socket programming in C available 13th January, due 20th January
    - Java RMI programming available 20th January, due 27th January
    - The Globus toolkit available 27th January, due 3rd February
  - 1 large programming assignment, worth 30% of total
    - Available 3rd February, due 10th March
    - Mark for the large programming assignment will be *multiplied* by the each of the marks gained in the 3 small exercises
  - All coursework to be handed in by 5pm on the due date
    - Hard deadlines: late submissions will receive zero marks unless valid special circumstances form submitted

Copyright © 2006 University of Glasgow

Questions on Module Structure?

Copyright © 2006 University of Glasgow

A Reference Model for Grid Computing

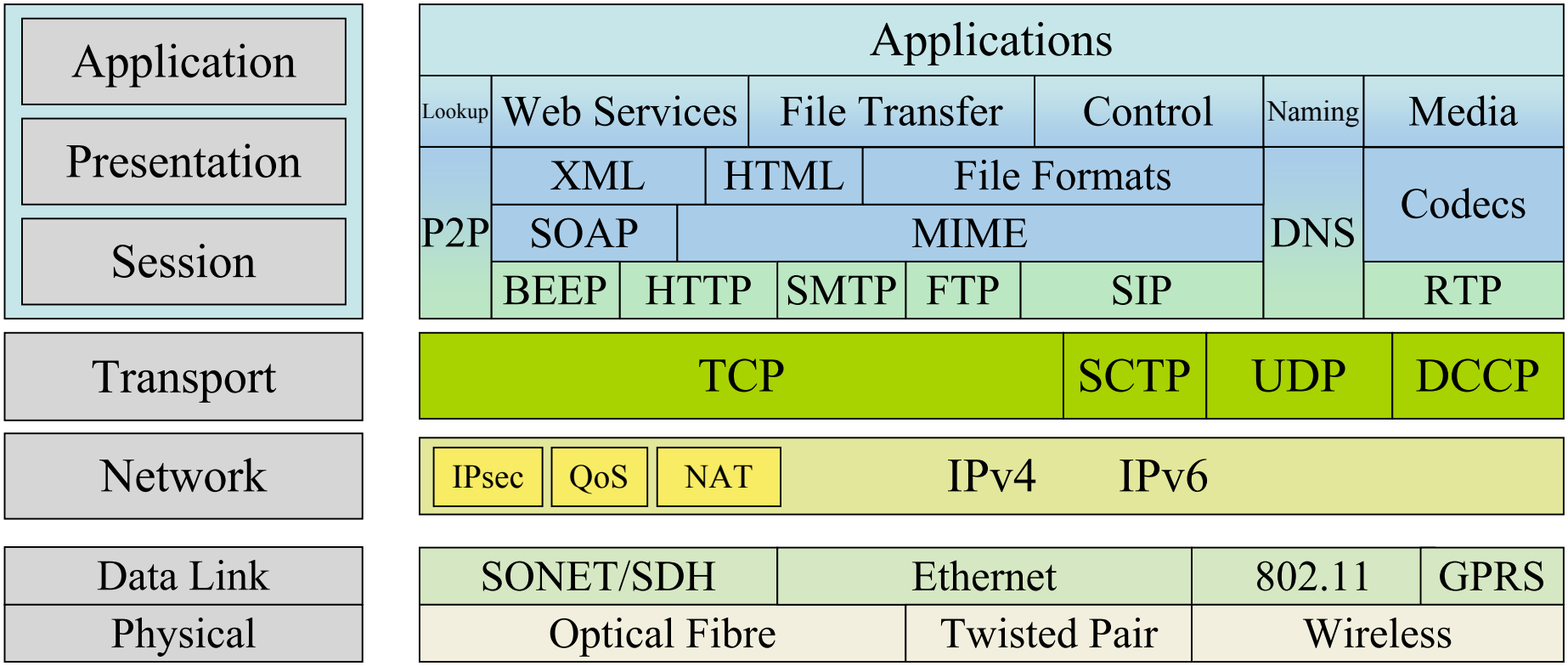


- Network protocols and applications follow a layered architecture
- Reference model for Grid Computing differs from the OSI model

⇒ Conceptual models – don't necessarily reflect reality

Copyright © 2006 University of Glasgow

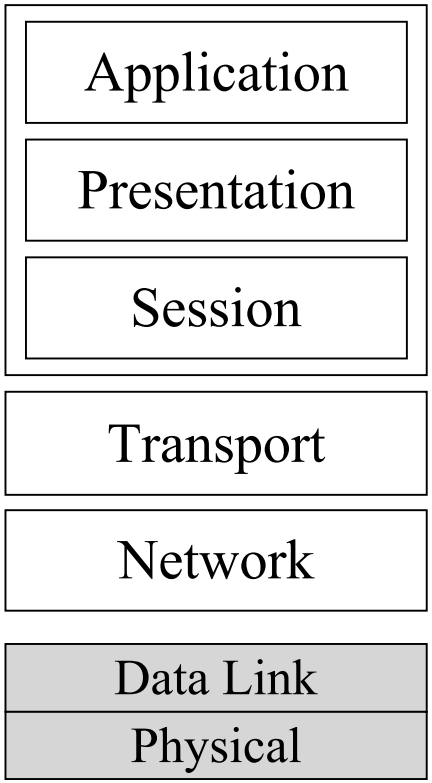
Implementation of Grid Computing

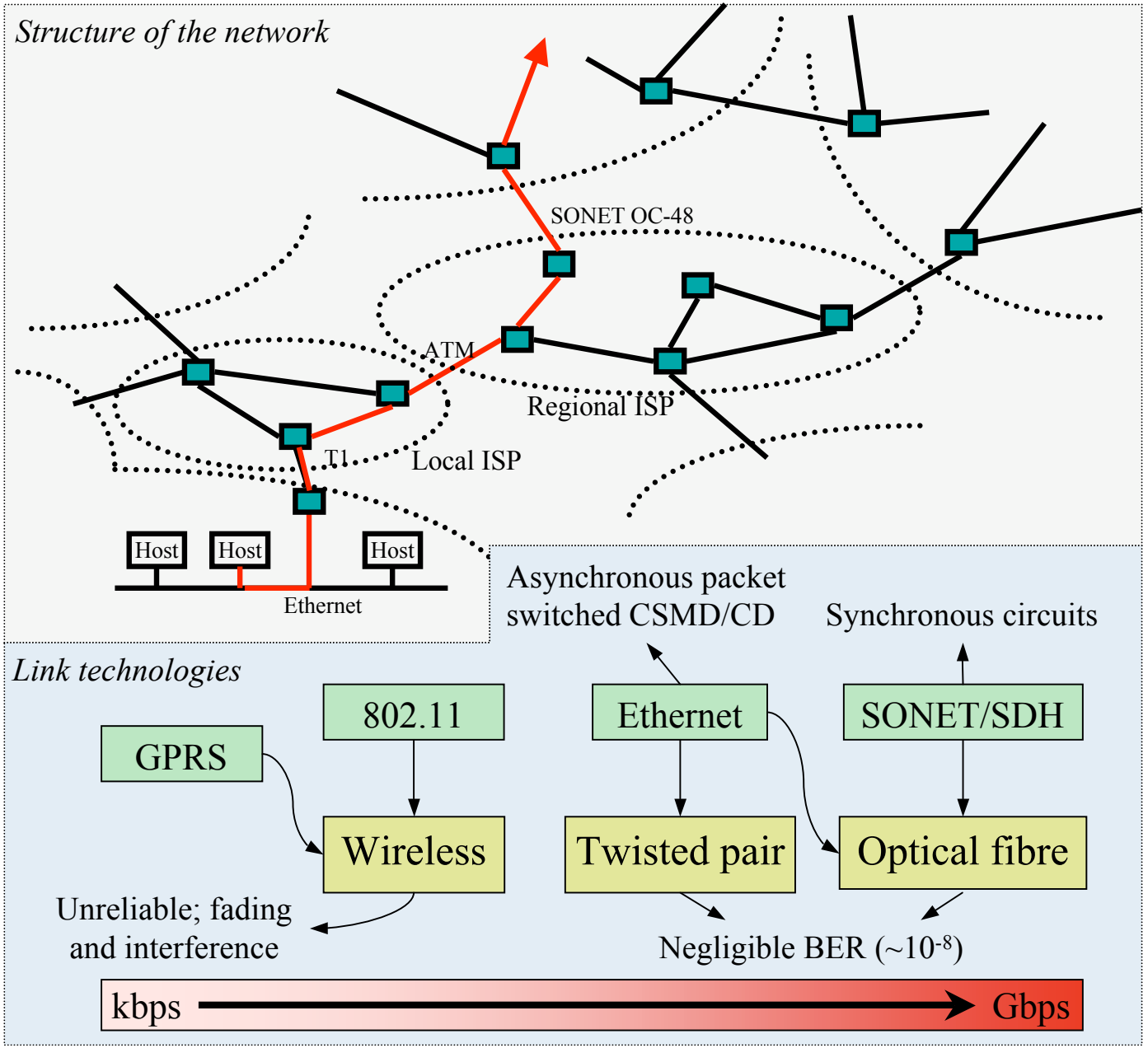


- No single "Grid Computing" protocol
  - A range of existing protocols, frameworks, standards, procedures, systems
  - Combine in novel ways; build large-scale heterogeneous applications
- Review behaviour of each layer before discussing system aspects

Copyright © 2006 University of Glasgow

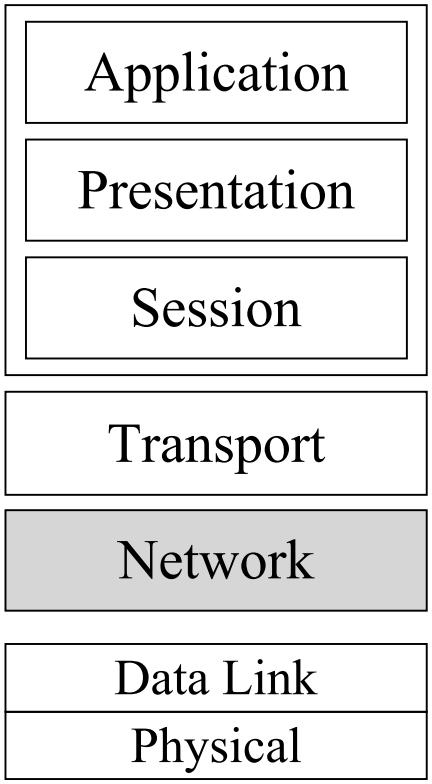
Link Layer Protocols





Copyright © 2006 University of Glasgow

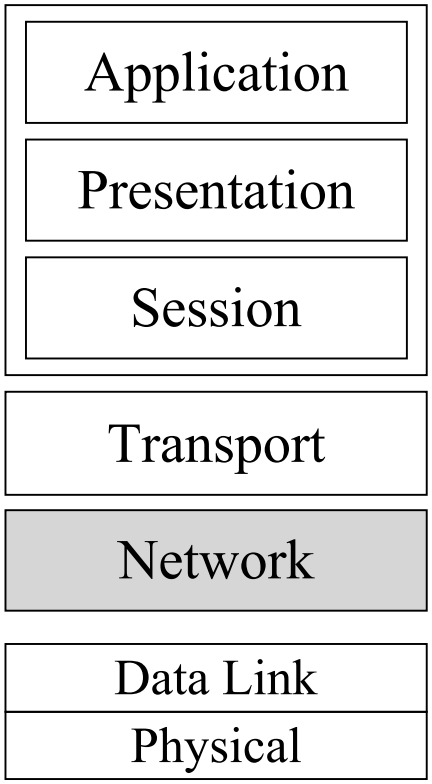
Network Protocols

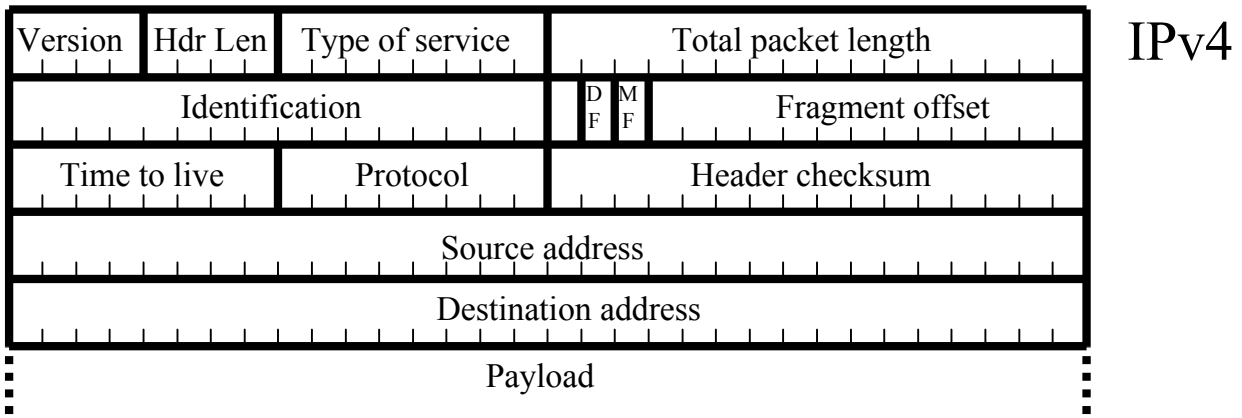


- Tie the disparate link layers together to form a single coherent network
  - Naming and addressing
  - Data delivery and quality of service
    - Packets vs. circuits vs. cells
  - Routing
  - Security
- IP dominates as inter-domain network protocol
  - IPv4 + NAT ⇒ IPv6
  - Others exist:
    - ATM ⇒ MPLS ⇒ MPλS
    - PSTN + ISDN
    - …

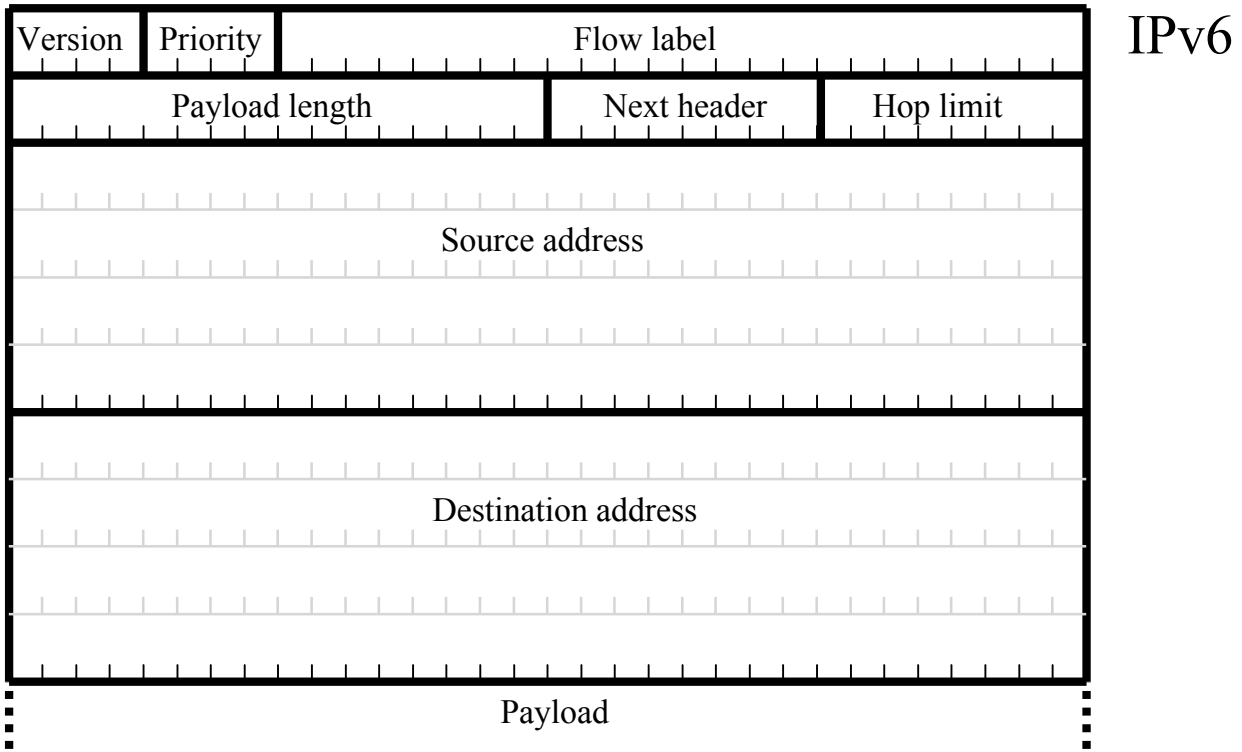
Copyright © 2006 University of Glasgow

Network Protocols: IPv4 and IPv6





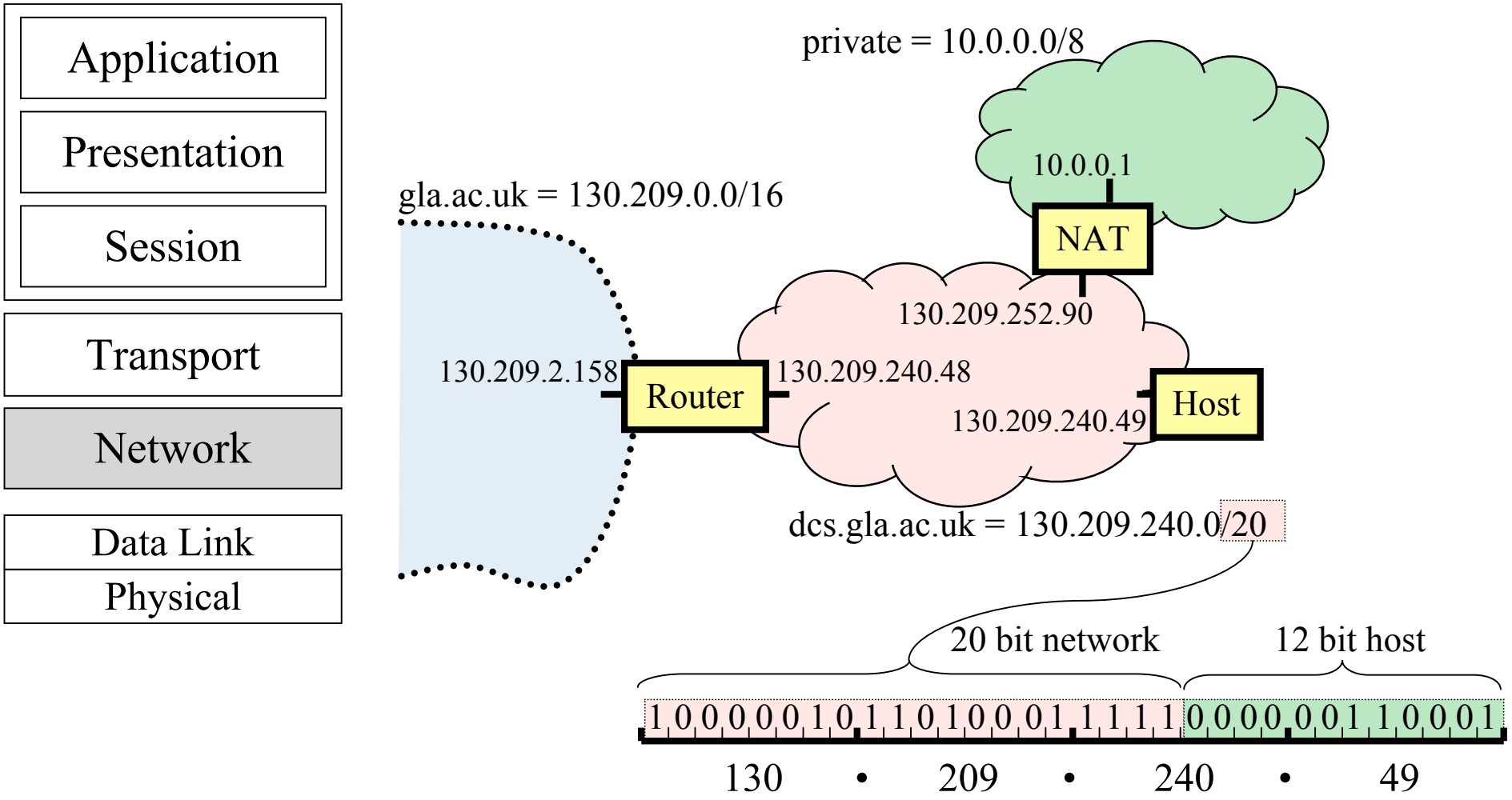
IPv4



IPv6

Copyright © 2006 University of Glasgow

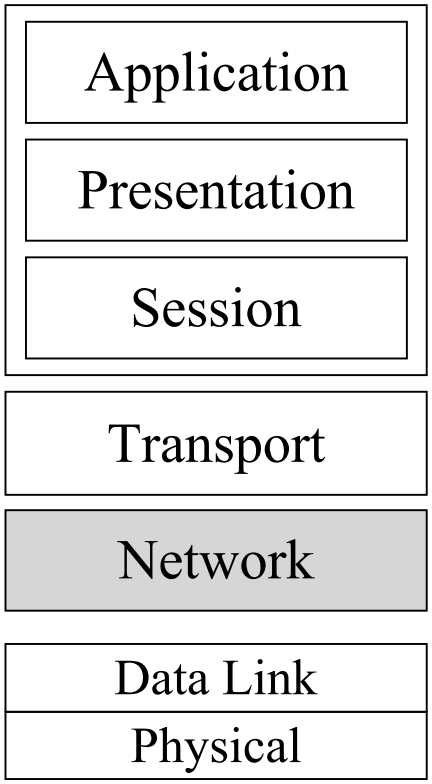
Network Protocols: IP Addressing

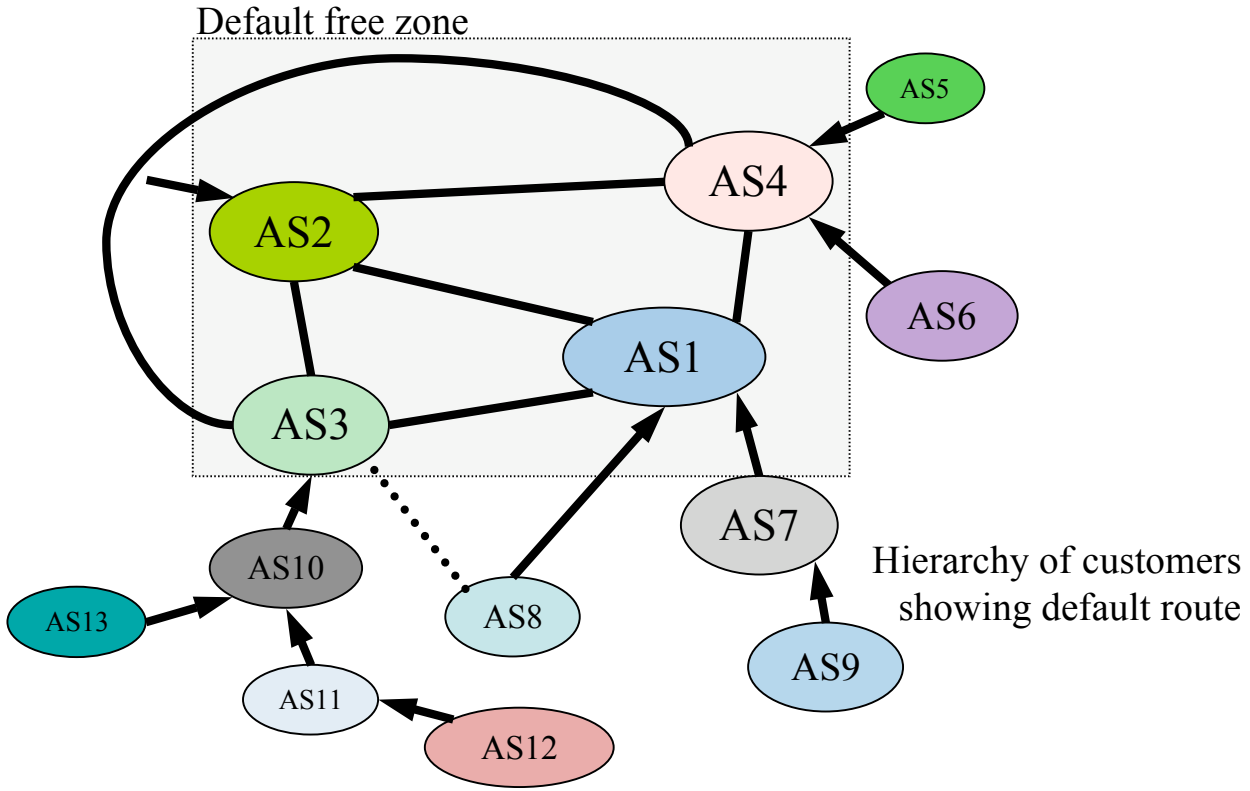


- IP address split into network and host parts
- Classless aggregation of networks
- Issues with NAT and use of private address space

Copyright © 2006 University of Glasgow

Network Protocols: Routing

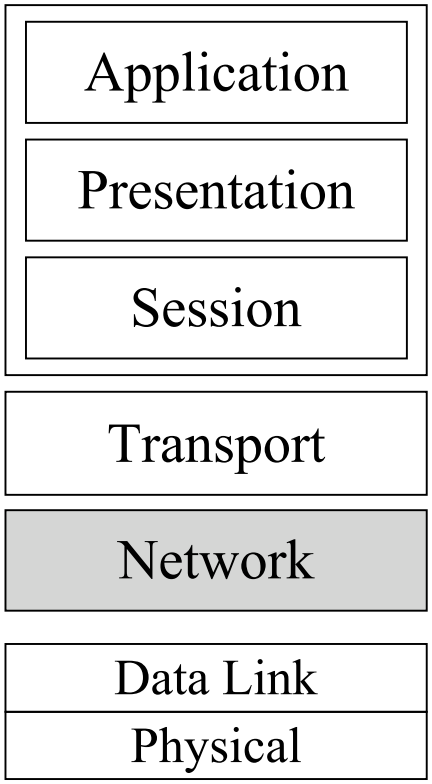




- Intra-domain shortest path routing (link state or distance vector)
- Inter-domain routing using path vector of autonomous systems
- Tier-1 default free zone with large hierarchy of customers' AS
- Scalability relies on IP address aggregation to hide leaf nodes → Mobility, Multi-homing
- Routing policy applied by filtering/adjusting paths

Copyright © 2006 University of Glasgow

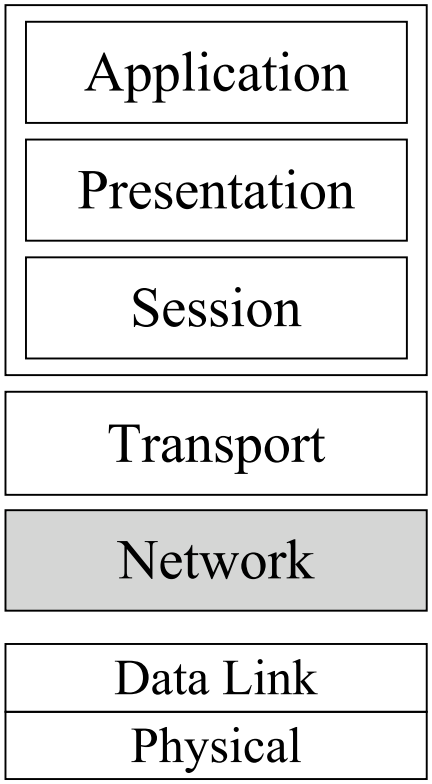
Network Protocols: Best Effort Service



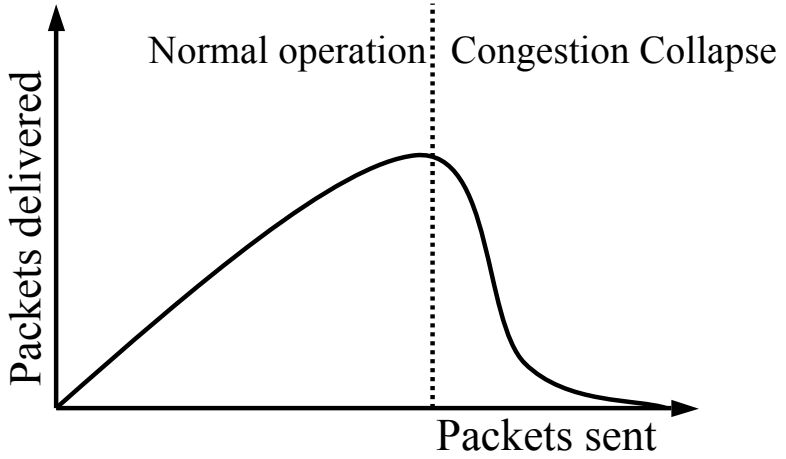
- IP provides a best effort packet delivery service
  - Packets may be lost, delayed, reordered or duplicated by the link layer; IP reflects this
  - IP layer will discard packets due to congestion at the output link from a router
  - Higher layer protocols compensate
    - Isolate or expose network timing behaviour?
    - Decouple reliability from that of the network?
  - Network engineering compensates
    - Packet loss, latency and jitter can be kept small through careful engineering and over-provisioning
    - Most backbone networks have very good performance
      - Essentially no loss
      - Very little queuing delay
    - Interconnects and customer LANs are currently the main trouble spots

Copyright © 2006 University of Glasgow

Network Protocols: Congestion and QoS



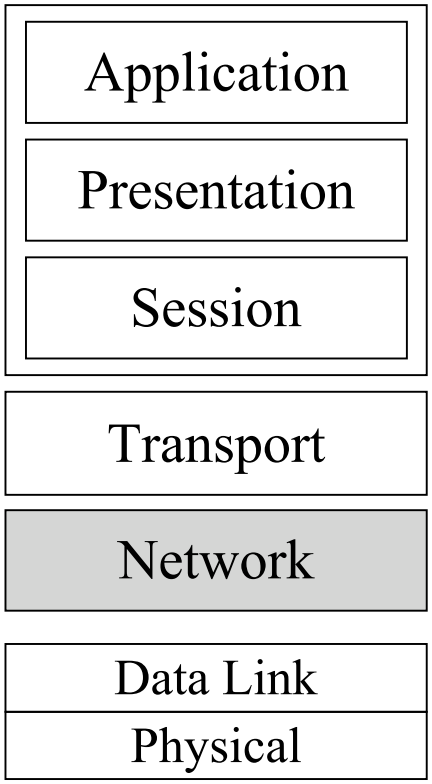
- Layers above IP expected to react to packet loss
  - As a signal to perform some loss recovery algorithm
    - Retransmission
    - Forward error correction
    - Loss tolerance
  - As a signal to reduce their sending rate
  - Congestion collapse may occur if higher layers ignore loss



  - Quality of service (QoS) protocols reserve capacity to allow some applications to avoid congestion control
    - Integrated services; differentiated services

Copyright © 2006 University of Glasgow

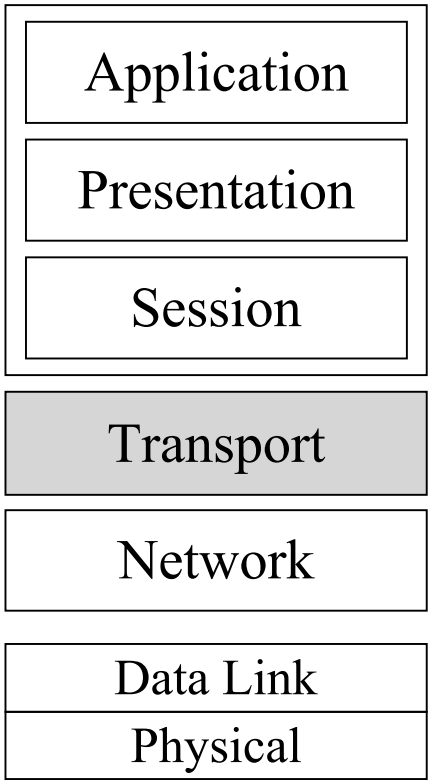
Network Protocols: Security



- IP layer typically operates without security
- Several types of security needed for robustness:
  - Host authentication and identification
  - Message confidentiality, non-repudiation, integrity protection and authentication
  - Routing security
  - Distributed denial-of-service attack prevention
- IPsec framework solves some of these
  - Not clear IP level appropriate place to solve
  - End-to-end principle applies

Copyright © 2006 University of Glasgow

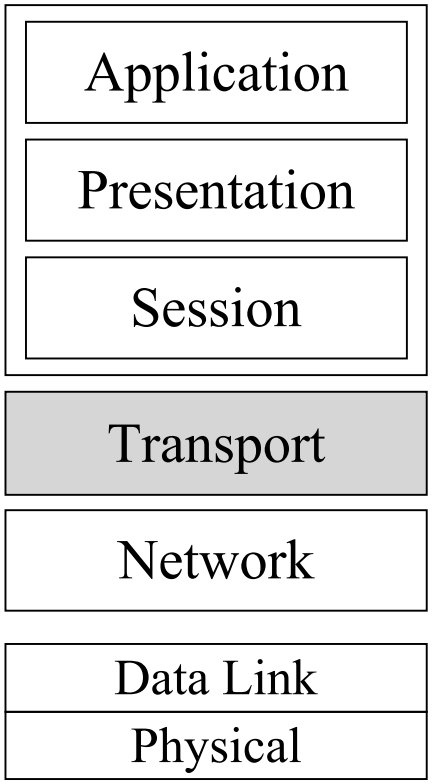
Transport Protocols

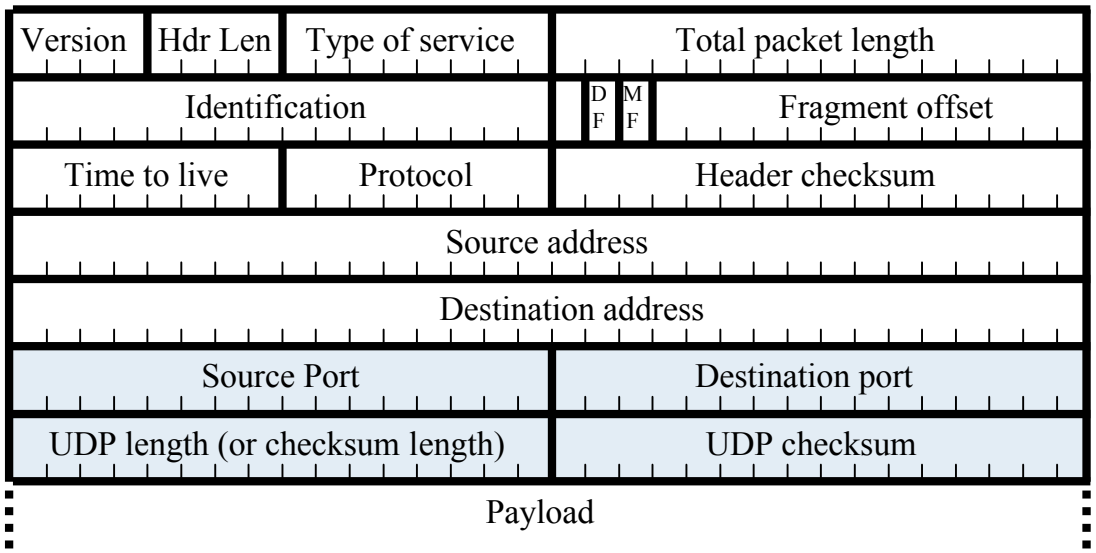


- The IP service, by itself, is very limited
  - Just (tries to) deliver packets
- Always augmented by a transport protocol
  - UDP/IP
  - TCP/IP
  - DCCP/IP
  - SCTP/IP
- The transport protocol will impact perceived reliability and timing performance of network

Copyright © 2006 University of Glasgow

Transport Protocols: UDP

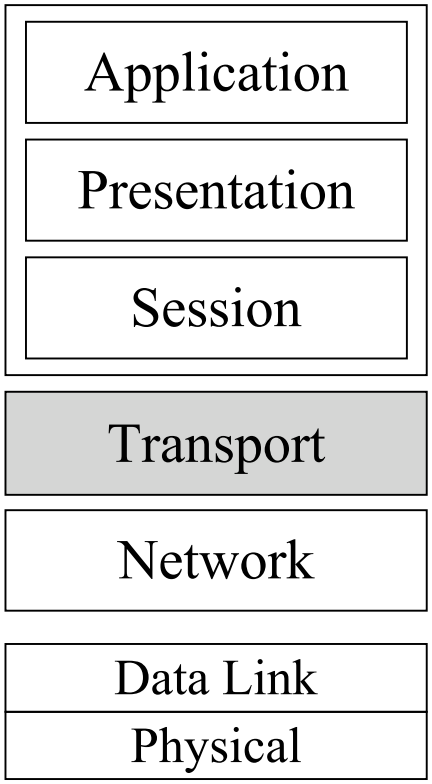




- Exposes the raw IP service to higher layers
  - Best effort (unreliable) connectionless packet delivery
    - Packet loss and jitter
    - Unicast and multicast
    - No congestion control
- Adds ports and payload checksum
  - UDP-lite overloads length field for partial checksum

Copyright © 2006 University of Glasgow

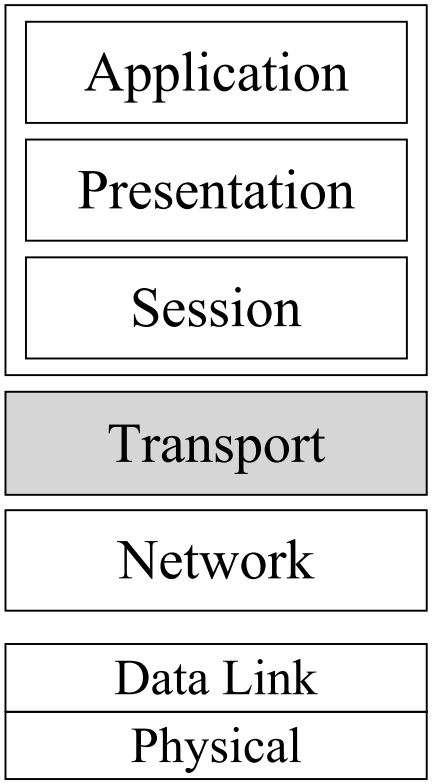
Transport Protocols: DCCP



- UDP service very limited
  - No association between sender and receiver
    - NAT binding difficult to setup, need application level refresh to avoid timeout
  - No congestion control
    - Leading to potential congestion collapse of network
      - e.g. multimedia applications using UDP
  - No abstraction
    - Complexity for application programmers
- DCCP newly standardised alternative
  - Unreliable, but connection-oriented and congestion controlled datagram service
  - E.g. for streaming multimedia

Copyright © 2006 University of Glasgow

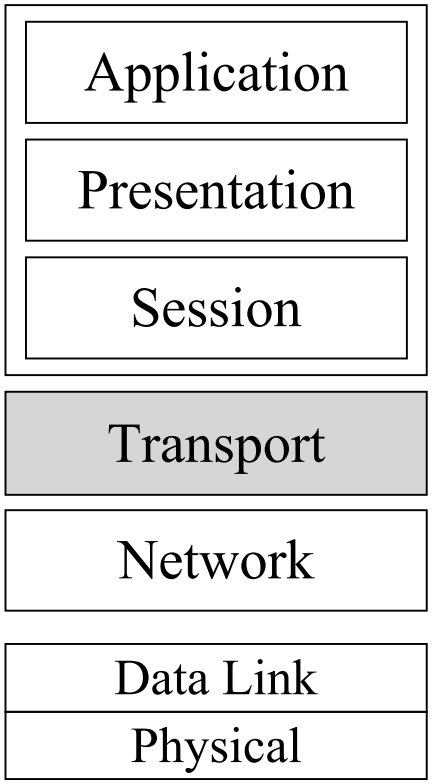
Transport Protocols: TCP



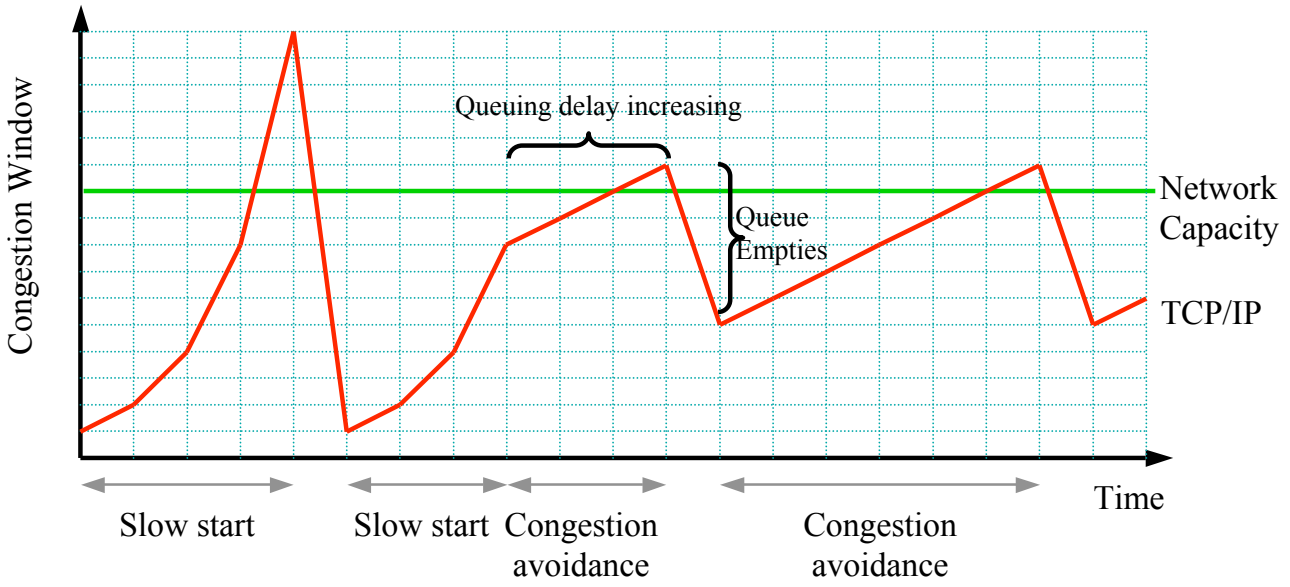
- Many systems need reliable data transfer ⇒ TCP
- Connection oriented, reliable and rate adaptive
  - Initial 3 way handshake
    - Connection setup latency/overhead
  - Reliable data transmission
    - Each packet contains a sequence number
    - Acknowledgements sent as packets arrive
    - Sender retransmits any lost packets
    - Receiver buffers data until all preceding packets have arrived, and presents to the application in order
- Abstracts complexity of the network
  - Provides reliable, not timely, byte stream service
  - Simple interface for application programmers

Copyright © 2006 University of Glasgow

Transport Protocols: TCP

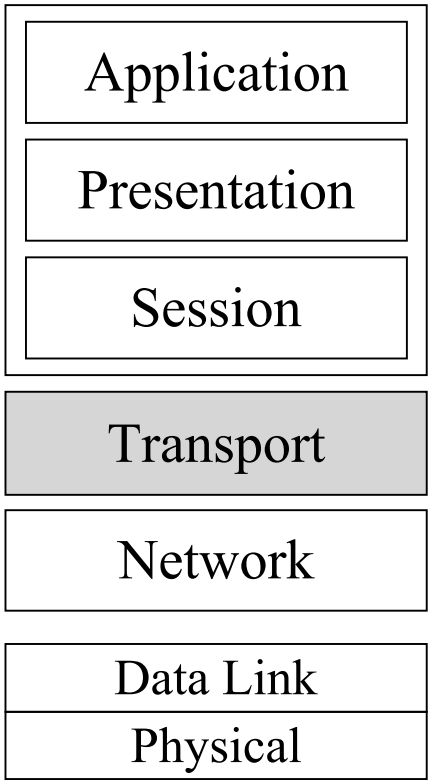


- Adapts sending rate to match network capacity
  - Window-based congestion control
    - Additive increase/multiplicative decrease
      - Linear probe of capacity until momentary overload
      - Multiplicative back-off to safe sending rate
  - High link utilization
  - Approximately fair share between flows

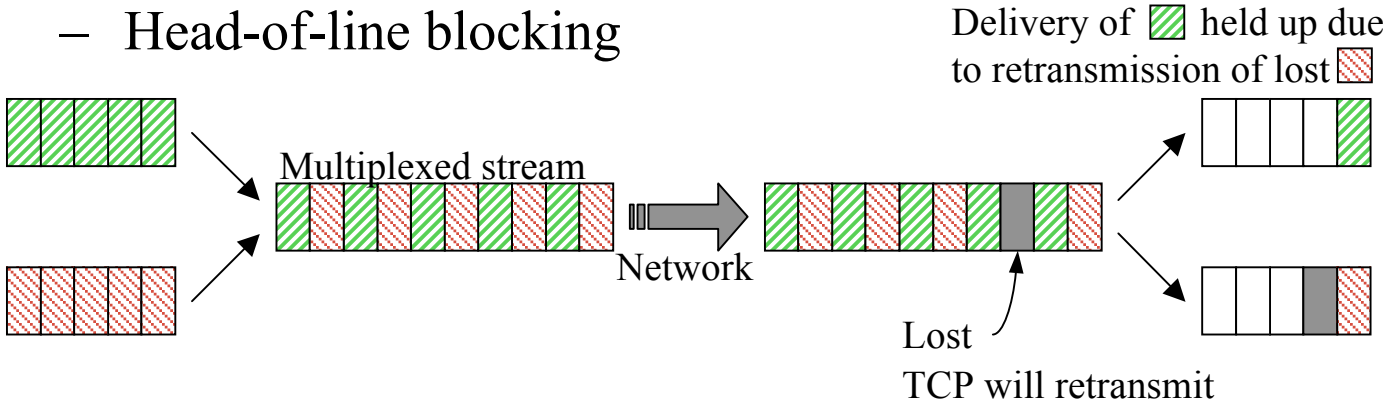


Copyright © 2006 University of Glasgow

Transport Protocols: SCTP



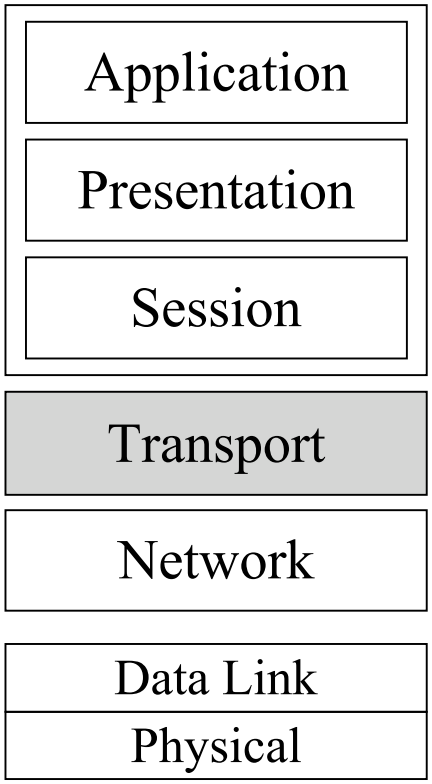
- TCP connections block and deliver data reliably and in order
- Difficult to multiplex application-level flows
  - No framing or record boundaries
  - Head-of-line blocking



- SCTP newly standardised alternative
  - TCP with explicit record boundaries, multiple flows within an association, flows do not block each other
  - e.g. for telephony signalling; web browsing

Copyright © 2006 University of Glasgow

The Berkeley Sockets Interface



- Standard API for low-level networking functions

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/inet.h>
#include <unistd.h>

int socket(int domain, int type, int protocol);

int connect(int s, struct sockaddr *name,
                                    socklen_t namelen);
ssize_t send(int s, void *msg, size_t len, int flags);

int bind(int s, struct sockaddr *name,
                                    socklen_t namelen);
int listen(int s, int backlog);
int accept(int s, struct sockaddr *addr,
                                    socklen_t addrlen);

FD_ZERO(&fdset);
FD_SET(fd, &fdset);
int select(int nfds, fd_set *r, fd_set *w, fd_set *e,
                                  struct timeval *timeout);
FD_ISSET(fd, &fdset);
ssize_t recv(int s, void *buf, size_t len, int flags);

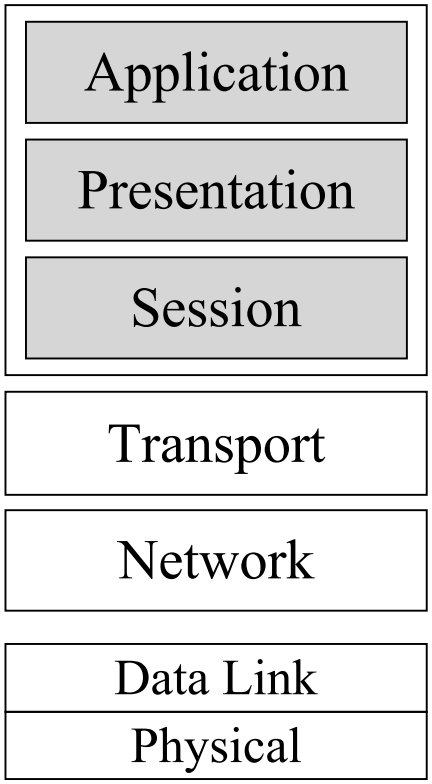
int close(int s);
```

AF_INET

SOCK_STREAM SOCK_DGRAM

Copyright © 2006 University of Glasgow

Higher Level Protocols



- Many higher layer protocols are used in Grid Computing
  - SMTP } + MIME
  - HTTP }
  - SOAP + XML web services
  - NFS and other RPC services
  - SIP + RTP
- Build upon the transport to provide more abstract services to applications

- Next few lectures review those used in traditional distributed systems and web services

Copyright © 2006 University of Glasgow

Summary

- What is grid computing?
- Module structure and administration
- Networking and communications
  - Layered protocol architectures
  - Review of IP networks and protocols
  - The Berkeley sockets interface

Copyright © 2006 University of Glasgow