Grid Computing: Networking and Communications

Grid Computing (M)

Lecture 1

UNIVERSITY *of* GLASGOW

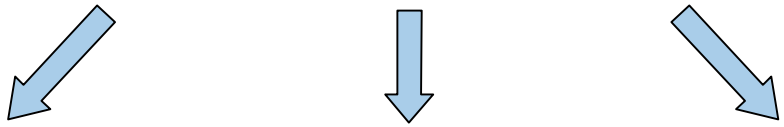
Lecture Outline

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

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

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

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

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, AC4 & DBIT4

The introductory lectures – starting today – will rapidly revise this material 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

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

Timetable

Note: tutorials take place in room 246B in the Kelvin Building

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

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 (1)	(csp)
	Lecture 8	Large Scale Systems Architecture (2)	(csp)
	Tutorial 1	Introduction to Globus	(ros)
Week 4	Lecture 9	Security (1)	(ros)
	Lecture 10	Security (2)	(ros)
	Tutorial 2	Developing with Globus	(ros)
Week 5	Lecture 11	Resource Management (1)	(pd)
	Lecture 12	Resource Management (2)	(pd)
	Tutorial 3	Globus Security	(ros)

Timetable

Week 6	Lecture 13	Resource Management (3)	(csp)
	Lecture 14	Resource Management (4)	(csp)
	Tutorial 4	Introduction to Condor	(ros)
Week 7	Tutorial 5	Permis and SAML	(ros)
	Tutorial 6	Portal Technologies	(ros)
	Tutorial 7	Q&A on programming assignment	(ros)
Week 8	Lecture 15	Scalability and Heterogeneity (1)	(joe)
	Lecture 16	Scalability and Heterogeneity (2)	(joe)
	Tutorial 8	OGSA-DAI	(ros)
Week 9	Lecture 17	Scalability and Heterogeneity (3)	(joe)
	Lecture 18	Systems Performance Evaluation (1)	(femi)
	Lecture 19	Systems Performance Evaluation (2)	(femi)
Week 10	Tutorial 9	Example systems	(ros)
	Lecture 20	Review and Future Directions	(csp)
	Tutorial 10	Assignment Demonstrations	(ros)

Lecturers

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

Peter Dickman
pd@dcs.gla.ac.uk

Joe Sventek
joe@dcs.gla.ac.uk

Richard Sinnott
r.sinnott@nesc.gla.ac.uk

Olufemi Komolafe
femi@dcs.gla.ac.uk

- Taught by a large team to cover the broad range of the subject
- Appointments to discuss the module or answer questions should be made by email

Reading List

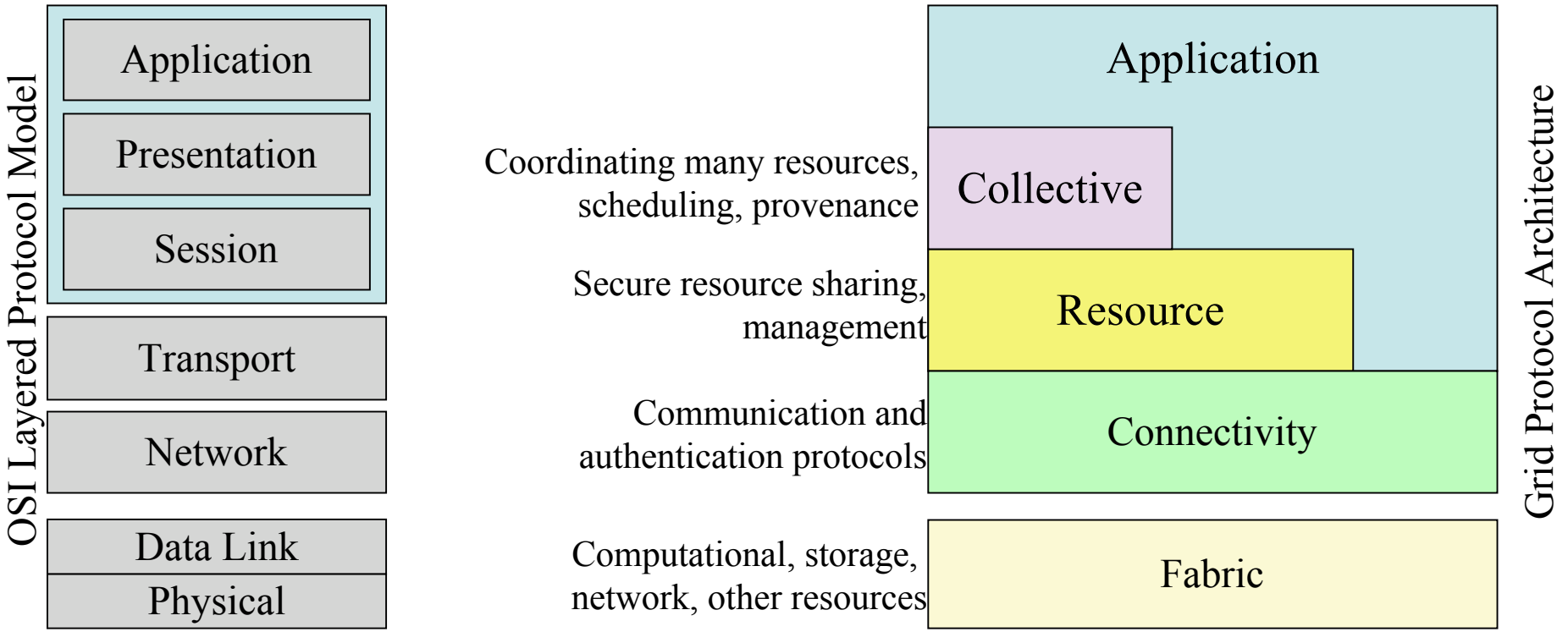
- There is no set text for this module. Research papers will be distributed as required, and technical manuals and related documentation will be issued as part of the practical activities.

http://www.dcs.gla.ac.uk/~csp/teaching/grid/

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 warm-up and revision exercises, each marked 0 or 1:
    - Socket programming in C available 12 January, due 5:00pm on 19 January
    - Java RMI programming available 19 January, due 5:00pm on 26 January
    - The Globus toolkit available 26 January, due 5:00pm on 2 February
  - 1 large programming assignment, worth 30% of total mark
    - Available 2 February, due 5:00pm on 9 March
    - Mark for the large programming assignment will be *multiplied* by the each of the marks gained in the 3 small exercises
  - Hard deadlines: late submissions will receive zero marks unless valid special circumstances form submitted

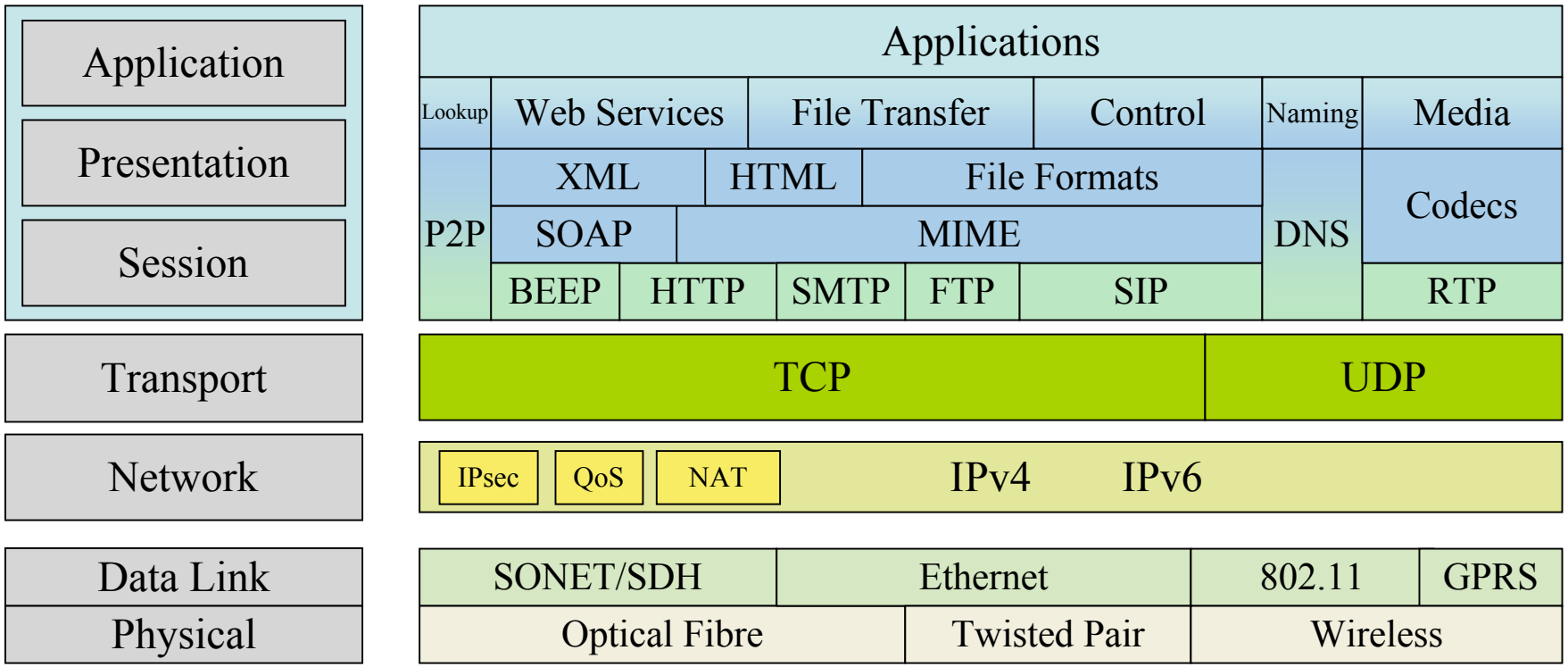
A Reference Model for Grid Computing



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

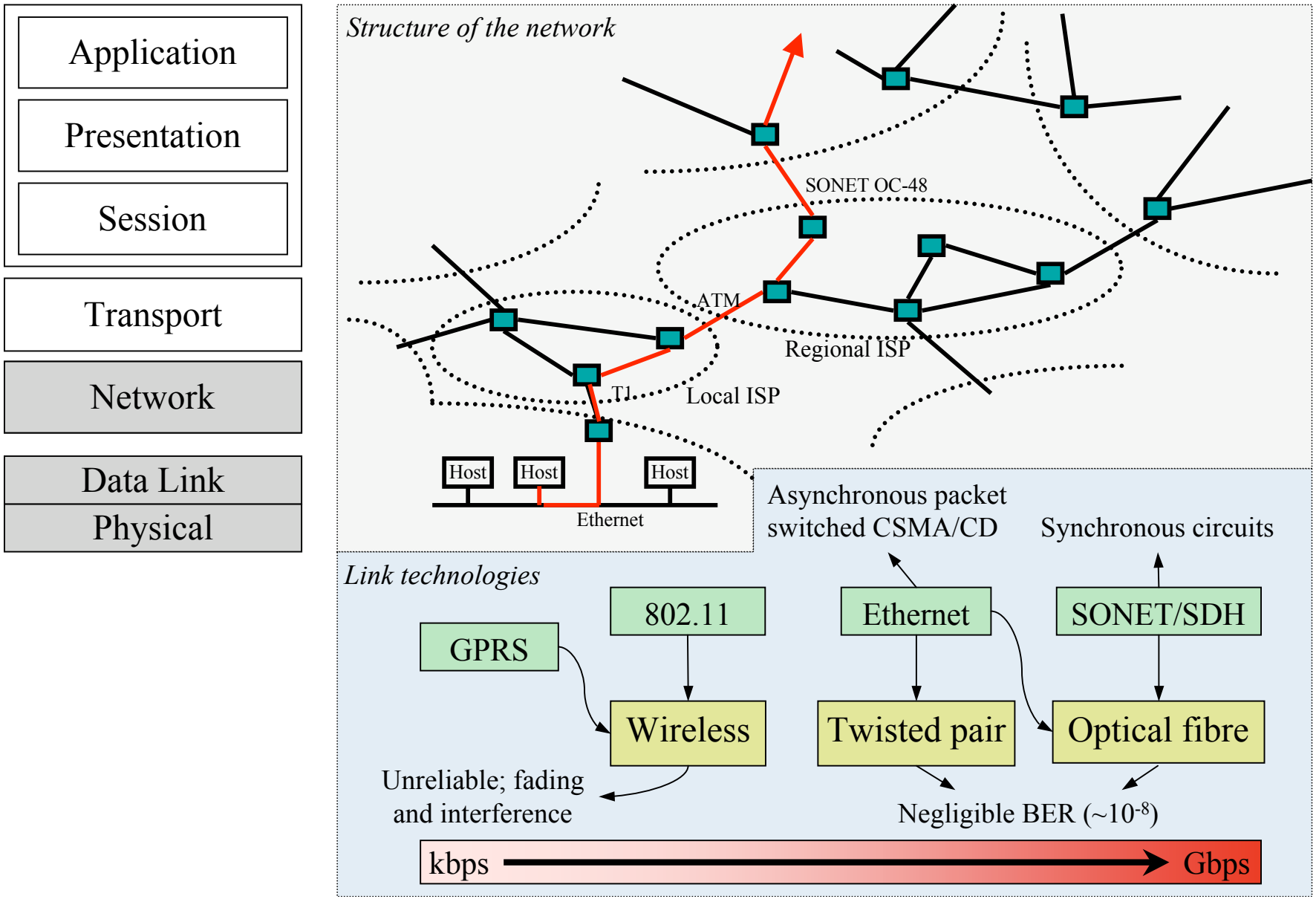
⇒ Conceptual models – don't necessarily reflect reality

Implementation of Grid Computing

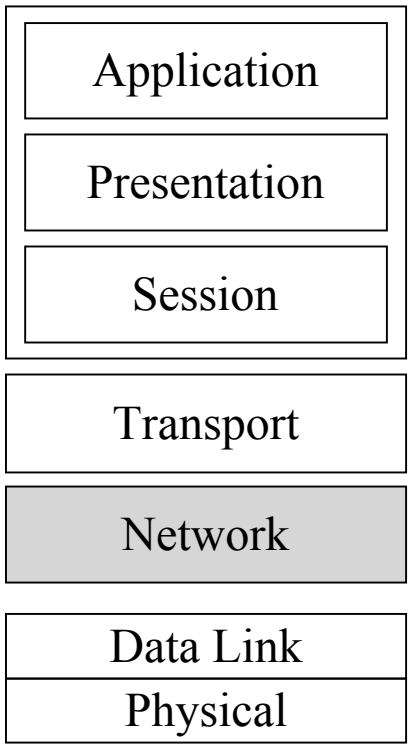


- No single "Grid Computing" protocol
  - A range of existing protocols, frameworks, standards, procedures, systems
  - Combine in novel ways; build large-scale heterogeneous applications
  - Must first understand the structure and properties of the network...

Structure and Properties of the Network



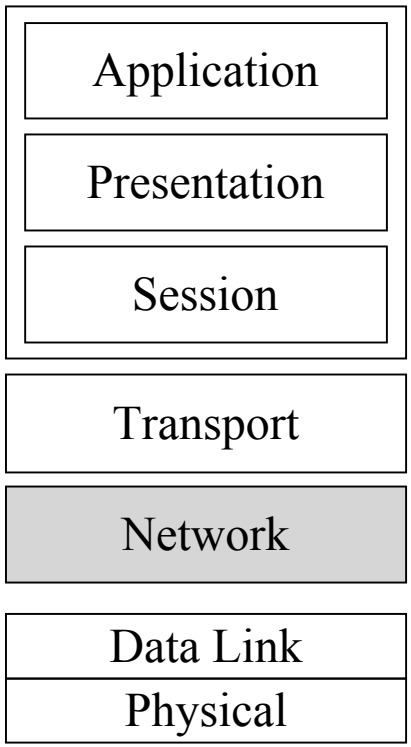
Network Protocols

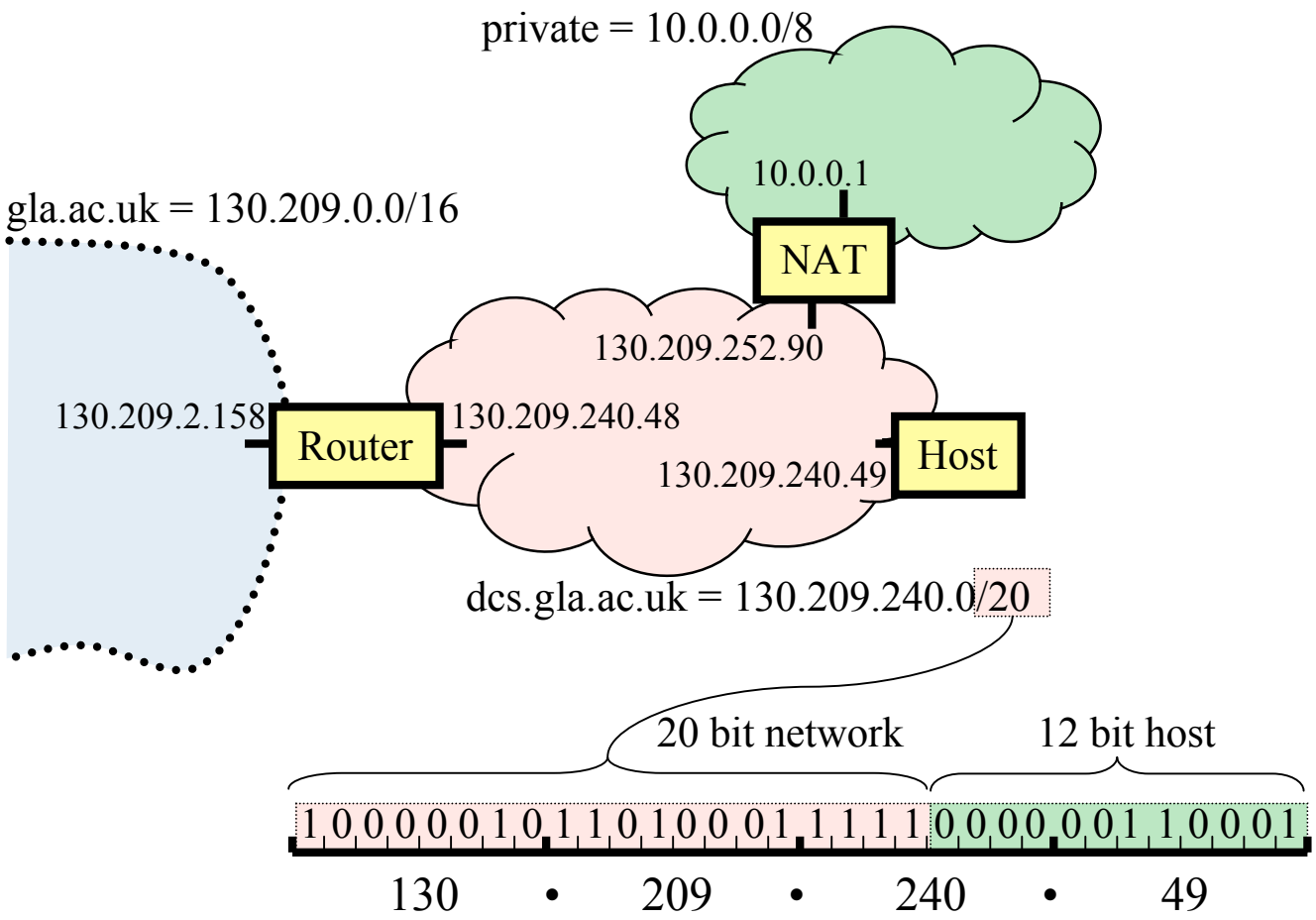


- Tie the disparate link layers together to form a single coherent network
  - Addressing & routing
  - Data delivery and quality of service
    - Packets *vs*. circuits *vs*. cells
    - Congestion control and quality-of-service
  - Transport protocols

- IP dominates as inter-domain network protocol
  - IPv4 + NAT ⇒ IPv6
  - Others exist:
    - ATM ⇒ MPLS ⇒ MPλS
    - PSTN
    - …

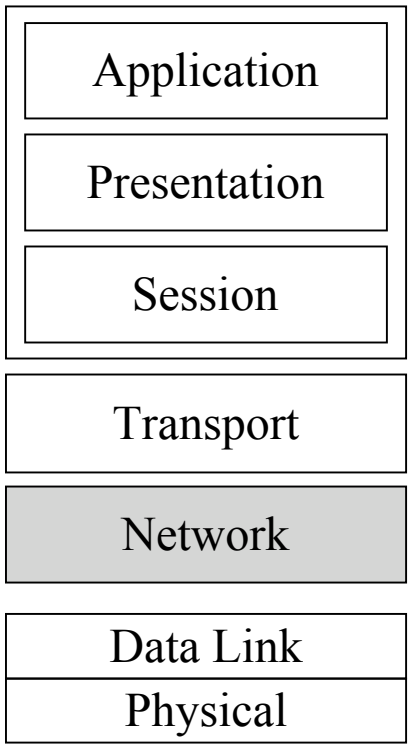
Network Protocols: Addressing & Routing

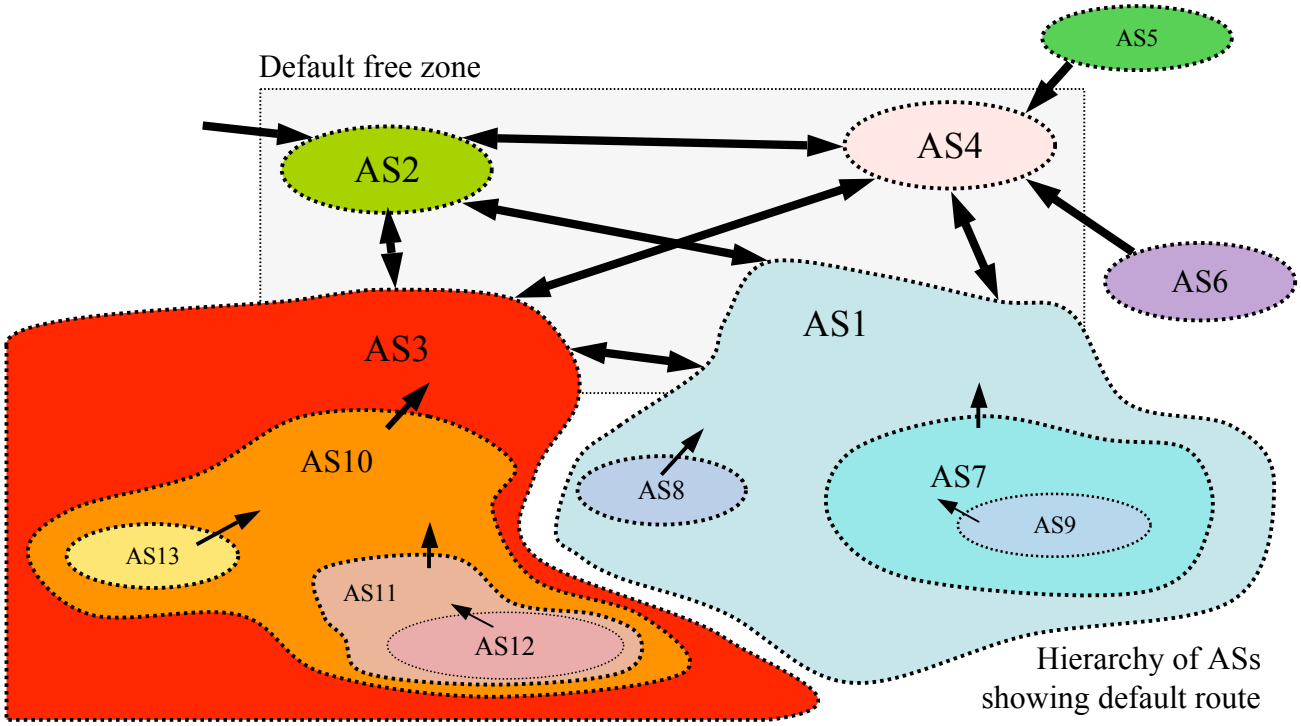




- IP address split into network and host parts
- Classless aggregation of networks
- Issues with NAT and use of private address space

Network Protocols: Addressing & Routing

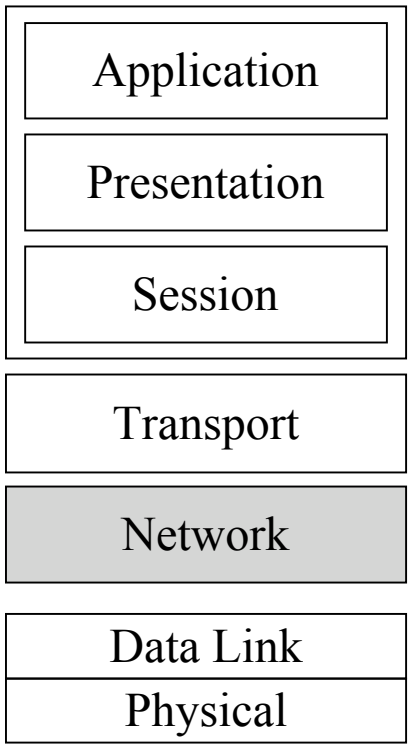




- Internet comprises a number of autonomous systems (ASs) each owning a part of the address space
- Inter-domain routing uses an AS path vector (BGP) to route to an address prefix

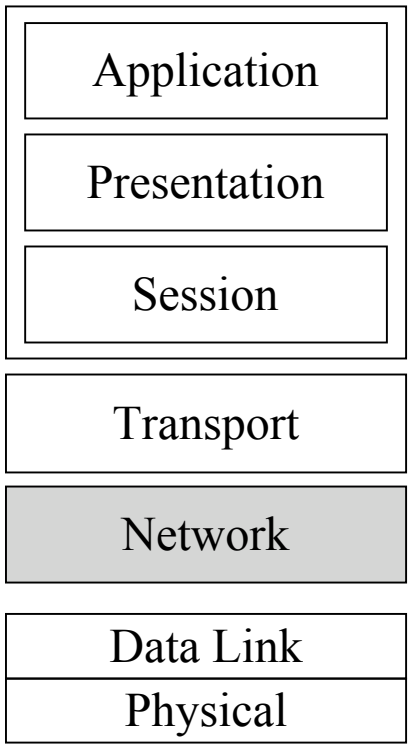
- Default-free zone in the core
  - Other ASs should use prefixes assigned from within provider address space
  - Relies of aggregation to scale
  - Issues due to autonomy and control

Network Protocols: Data Delivery & QoS

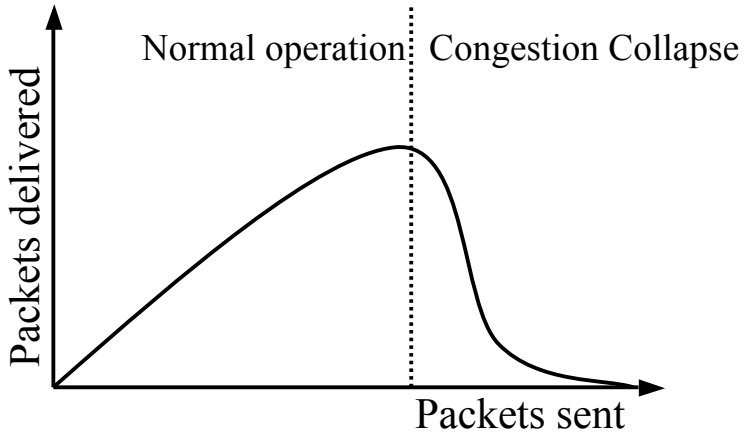


- IP provides a best effort packet delivery service
  - Packets may be lost, delayed, reordered or duplicated by the link layer; IP reflects this
  - IP layer will discard packets due to congestion at the output link from a router
  - Network engineering compensates
    - Packet loss, latency and jitter 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

Network Protocols: Data Delivery & QoS

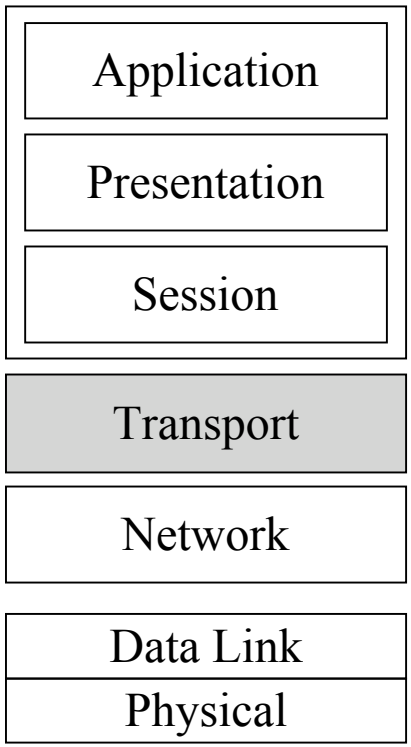


- Layers above IP expected to react to packet loss
  - As a signal to perform some loss recovery algorithm
    - Retransmission
    - Forward error correction
    - Loss tolerance
  - As a signal to reduce their sending rate
  - Congestion collapse may occur if higher layers ignore loss



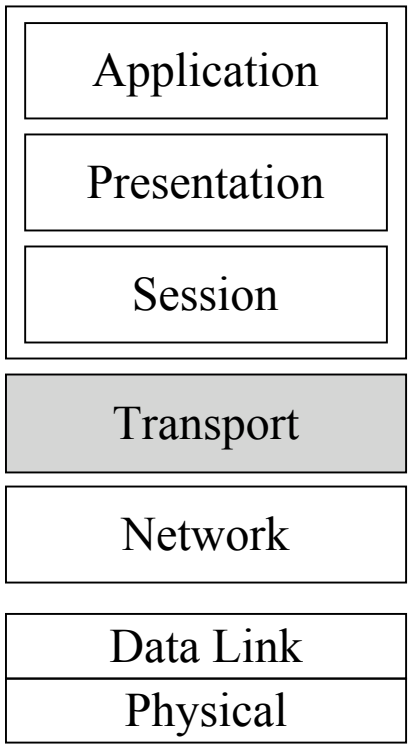
  - Quality of service (QoS) protocols reserve capacity to allow some applications to avoid congestion control
    - Integrated services; differentiated services

Transport Protocols



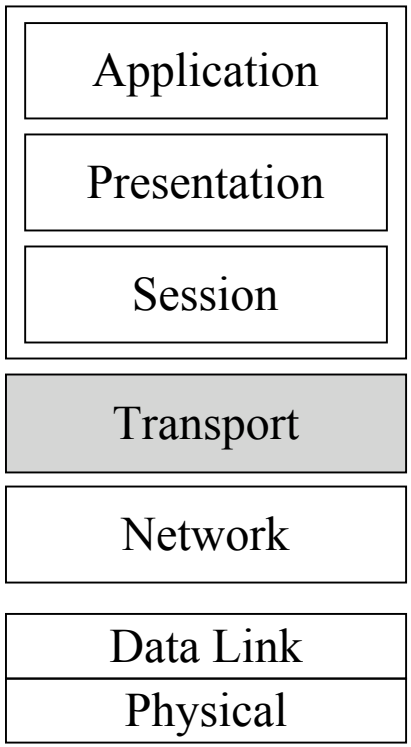
- The IP service, by itself, is very limited
  - Just (tries to) deliver packets
- Always augmented by a transport protocol
  - UDP/IP
  - TCP/IP
- The transport protocol will impact perceived reliability and timing performance of network

Transport Protocols: UDP



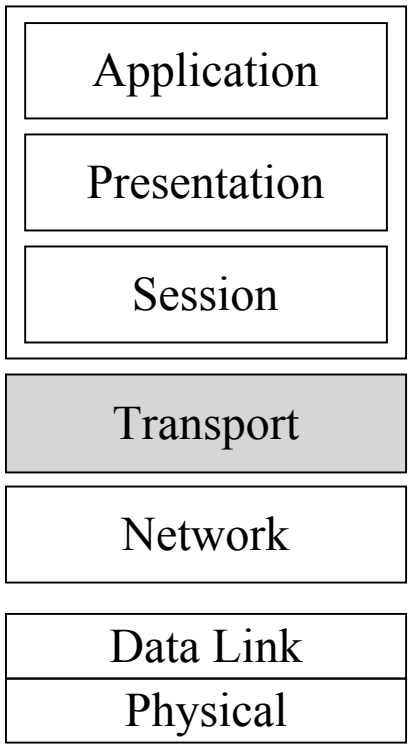
- Exposes the raw IP service to higher layers
  - Best effort (unreliable) connectionless packet delivery
    - Packet loss and jitter
    - Unicast and multicast
    - No congestion control

Transport Protocols: TCP

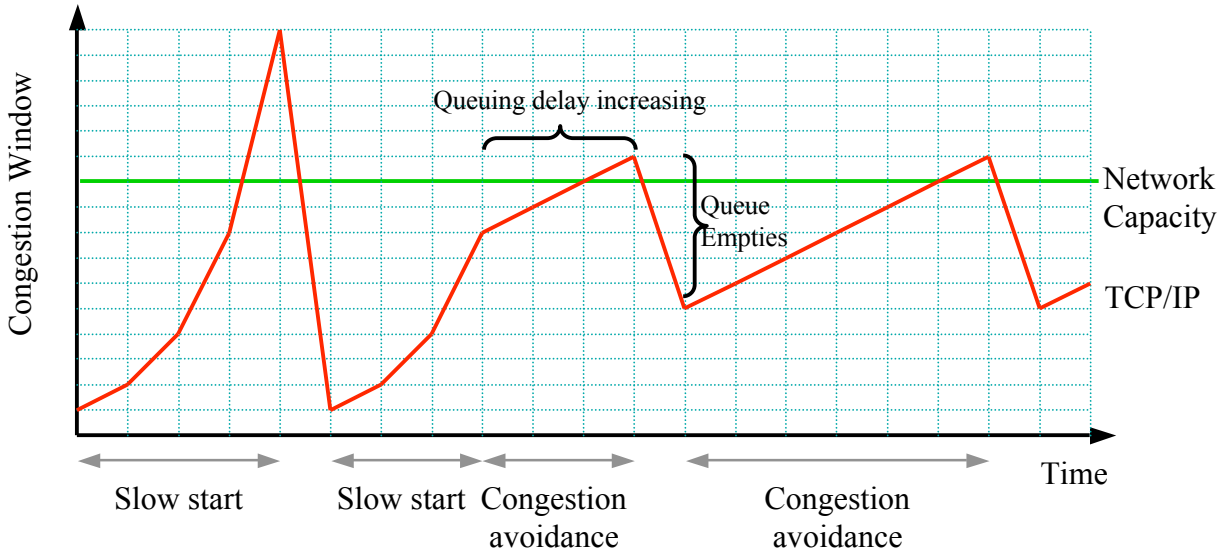


- Many systems need reliable data transfer ⇒ TCP
- Connection oriented, reliable and rate adaptive
  - Initial 3 way handshake
    - Connection setup latency/overhead
  - Reliable data transmission
    - Each packet contains a sequence number
    - Acknowledgements sent as packets arrive
    - Sender 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

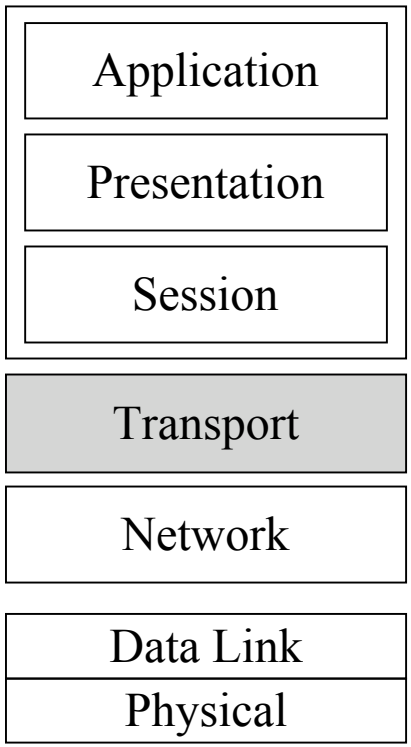
Transport Protocols: TCP



- 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 utilization at low speeds; problems at high speed
  - Approximately fair share between flows



Berkeley Sockets Interface



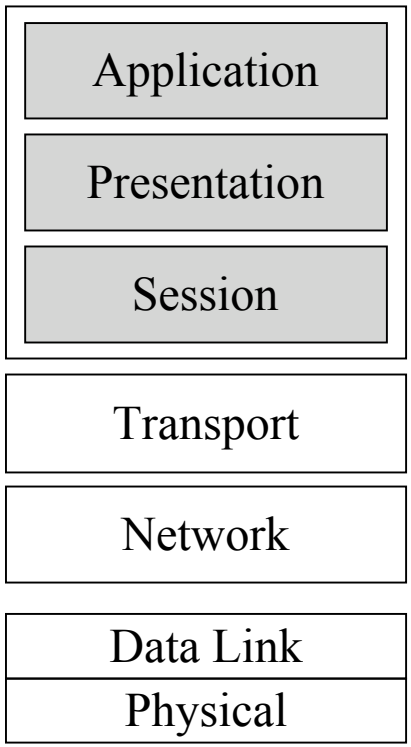
- Standard low-level API for networking functions

```
#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

Higher Level Protocols



- Many higher layer protocols are used in Grid Computing
  - SMTP
  - HTTP

    } + MIME
  - SOAP + XML web services
  - NFS and other RPC services
  - SIP + RTP
- Build upon the transport to provide more abstract services to applications
  - Hide the complexity of the sockets API in *middleware*
- Next few lectures review those used in traditional distributed systems and web services

Summary

- What is grid computing?
- Module structure and administration
- Networking and communications
  - Layered protocol architectures
  - Review of IP networks and protocols
  - Network APIs

Next lecture at 10:00am tomorrow, in Maths 325

Time change for this week only