

# CPP—Cello packet platform

Lars-Örjan Kling, Åke Lindholm, Lars Marklund and Gunnar B. Nilsson

CPP is a carrier-class technology that has been positioned for access and transport products in mobile and fixed networks. It is an execution and transport platform with specified interfaces for application design. The execution part consists of support for the design of application hardware and software. The transport part, which can be seen as an internal application on the execution platform, consists of several protocols for communication, signaling, and ET transmission. Typical applications on current versions of CPP include third-generation nodes—RBSs, RNCs, media gateways, and packet-data service nodes/home agents. CPP was first developed for asynchronous transfer mode ATM and TDM transport. Now, support is being added for IP transport.

The authors describe the technical and customer benefits of adding IP support in CPP, walking the reader through the basic principles for IP services in CPP and the CPP IP architecture, which is very robust and scalable.

## Why IP transport?

The world of telecommunications is characterized by change. Yesterday's dominant traffic type—voice—is being superseded by data traffic. In future telecommunications networks, voice will occupy only a small portion of bandwidth. This will put new demands on the networks, which will have to be packet-oriented and at the same time able to handle delay-sensitive traffic, such as voice and video (video conferencing). ATM technology has long been considered the solution to quality of service (QoS) in networks, but several areas of concern have since been identified in large and growing networks. These are

- scalability—the cost of the extra ATM layer makes it difficult for ATM vendors and operators to increase the link speed above 1 Gbit/s;
- administration and maintenance—in large ATM networks, the number of permanent virtual circuits (PVC) for router interconnections increases significantly, and thus the administration and maintenance of these PVCs becomes a major issue when networks need to be upgraded. A similar problem exists in pure IP networks, but the addition of ATM does not simplify matters. In this sense, ATM is an extra layer that greatly increases complexity; and
- cost—for example, the cost of having to support and maintain two kinds of network equipment.

These issues have become a driving force for introducing IP routing into telecommunications networks.

The Internet boom has been accompanied by increased investments in IP technology, and a lot of effort has gone into solving quality-of-service and routing administration issues. Multiprotocol label switching (MPLS), differential services (DiffServ, Box B), multiclass extension (MCE), and header compression have made it possible to replace ATM with pure IP networks. And CPP is well positioned to handle the new environment (as well as the transition to it). DiffServ and MCE will be implemented in the first IP release of CPP, and the architecture is ready to make use of MPLS.

## BOX A, TERMS AND ABBREVIATIONS

API	Application program interface	PBA	Printed board assembly
ATM	Asynchronous transfer mode	PDSN	Packet-data service node
BGP	Border gateway protocol	PPP	Point-to-point protocol
CPP	Cello packet platform	PVC	Permanent virtual circuit
DiffServ	Differential services	QoS	Quality of service
E1/J1/T1	PDH transmission frame formats for 2 Mbit/s (E1) or 1.5 Mbit/s (J1/T1) transmission rates	RBS	Radio base station
ET	Exchange terminal	RIP	Routing information protocol
FIB	Forwarding information base	RNC	Radio network controller
HA	Home agent	RSVP-TE	Resource reservation protocol – traffic engineering
IP	Internet protocol	SCTP	Stream control transmission protocol
IPv4, IPv6	IP version 4, IP version 6	SIGTRAN	Signaling transport
MCE	Multiclass extension	SS7	Signaling system no. 7
MGW	Media gateway	STM-1	SDH transmission frame format for 155 Mbit/s
MPLS	Multiprotocol label switching	TCP	Transmission control protocol
O&M	Operation and maintenance	TDM	Time-division multiplexing
OC3	Optical carrier 3 (155 Mbit/s)	UDP	User datagram protocol
OSPF	Open shortest path first		

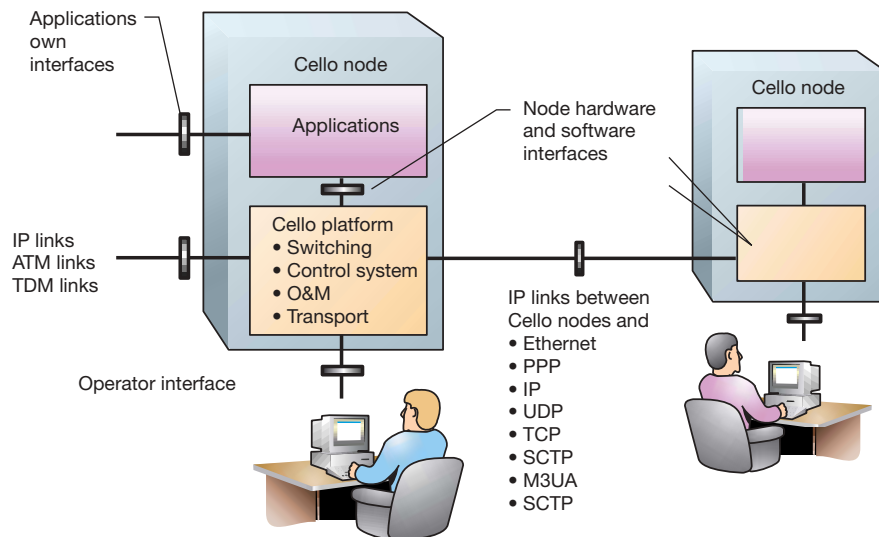


Figure 1  
A network view of the Cello packet platform (CPP) including transport protocols.

## Who are the customers?

The introduction of IP in telecommunications networks will commence in the core network in response to demands for large, high-speed networks. Datacom vendors are currently trying to grab as many market shares as they can. At present, these vendors can solely offer pure IP router products. Therefore, network operators can either opt

to integrate telecommunications equipment and IP routers themselves or they can buy complete site solutions from a telecommunications vendor. Over time, the IP migration will extend further and further from the core network into the access network.

Ericsson knows that operators want vendors to come up with solutions and migration stories that are cost-effective and easy to maintain (Figure 2). The introduction of IP in

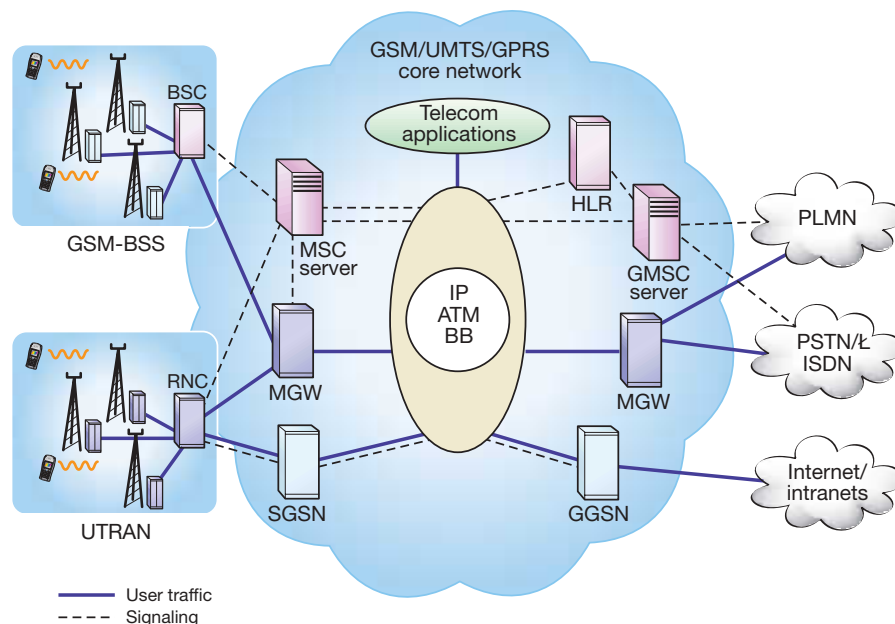
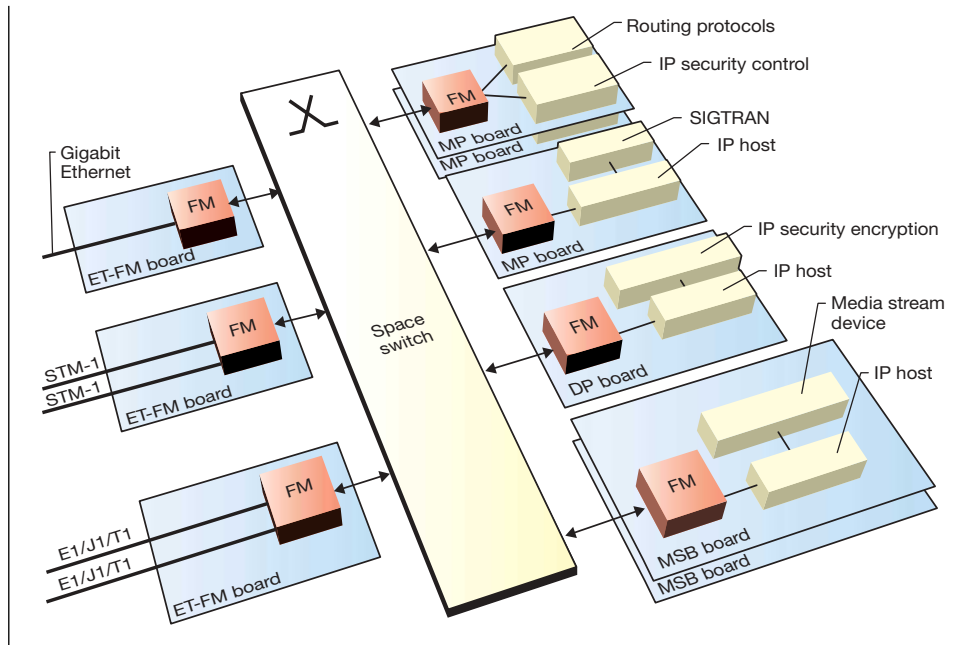


Figure 2  
A telecommunications network with different types of backbone and access networks.

Figure 3  
Embedded and distributed IP router.



### BOX B, DIFFERENTIATED SERVICES

Traditional routers serve packets on a first-come-first-served basis. Since no differentiation is made with respect to the actual demands put on the delivery times of packet flows, this is often called best-effort service. In a best-effort network, packet flows that carry delay-sensitive voice receive the same service as packet flows that carry, say, a file transfer.

Different approaches have been proposed to solve this shortcoming. According to one approach, called the *IntServe* model, a flow must reserve resources in the routers on the path before packets can be sent. If sufficient network resources are not available, the flow is denied. An implication of *IntServ* is that every router in the network must store a per-flow state. This was seen as a major drawback and led to the introduction of a lightweight model called differentiated services (*DiffServ*). The *DiffServ* model is based on the following basic principles:

- A limited number of service classes is defined for specific purposes. These classes differ in terms of maximum delay or maximum drop probability.
- At the edge of a network, packets are marked in the header to reflect the service class to which the packet flow belongs.
- The routers in the network serve the packets according to its service class.

In principle, packet flows that belong to the same service class are merged into a com-

mon aggregated flow. The routers see these aggregated flows, but not individual packet flows.

A packet enters a classifier, where the service class mark is inspected. Depending on the mark, the packet is sent to one of several queues via an *enqueueer*. Depending on the filling in the queue, the packet might be discarded by the enqueueer according to a buffer management algorithm. Using a scheduling algorithm, a scheduler fetches packets one-by-one from the queues.

The buffer management and scheduling algorithms are configured to guarantee a specific per-hop-behavior for each service class, of which there are three standard classes: One best-effort class (BE) and two assured-forwarding classes (AF). Each assured-forwarding class is guaranteed a certain minimum bandwidth at which the queues are served. Excess bandwidth is distributed to other classes. The best-effort class is served only if there are no packets in the other queues. If a queue is almost full, incoming packets to that queue are discarded in a pseudo-random fashion.

In terms of resources, the *DiffServ* model causes the network to behave as if a few different logical networks were separated from each other. For example, one logical network could be used for voice traffic and another for file transfers.

telecommunications networks adds yet another transport protocol. Today operators are struggling with the migration from TDM to ATM. Later, when operators introduce IP, they will have to consider even more complex networks. Ericsson can offer telecommunications nodes with built-in transport capabilities for TDM, ATM and IP. Operators can thus capitalize on their installed base, especially in mobile networks. Likewise, Ericsson knows that network operators want a compact and cost-effective solution that has a consistent network-management interface. Ericsson will thus make pure IP routers obsolete in operator networks. CPP is the ideal choice of carrier-class technology for all telecommunications nodes in the access network, including the edge nodes to the core backbone.

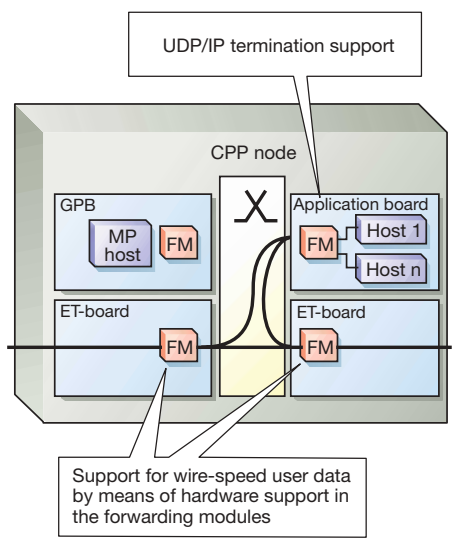


Figure 4 Fully distributed IP forwarding.

## Basic principles

Six basic principles for the IP services in CPP add value:

- embedded and distributed IP router by means of routing protocols in the main processor cluster and distributed forwarding on all device boards (Figure 3);
- fully distributed forwarding of IPv4 or IPv6 modules can be implemented in hardware or software. A hardware implementation can handle wire-speed transport (Figure 4);
- internal IP hosts can be located on a single printed board assembly (PBA) and connected to the IP router via the local forwarding module on the PBA (Figure 5). The IP host provides the application program interface (API);
- the internal hosts are IP end-systems that are visible and addressable from the external network and from within the node (Figure 5);
- a CPP node can house multiple virtual routers. In this case, each IP interface in the node is configured to belong to one specific virtual router; and
- CPP has built-in support for signaling system no. 7 (SS7) signaling gateway functionality by means of a complete SS7 stack for SIGTRAN, IP, ATM and TDM (Figure 6).

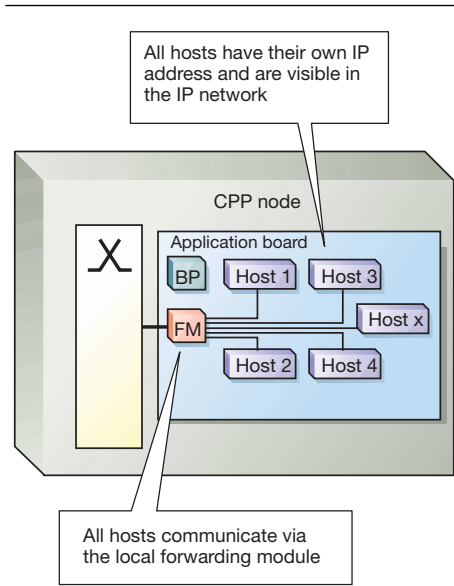


Figure 5 Multiple internal IP hosts.

## Architecture

The CPP IP architecture is composed of several subsystems that interwork with each other through well-defined interfaces (Figure 7). The IP forwarding subsystem provides fully distributed forwarding of IPv4

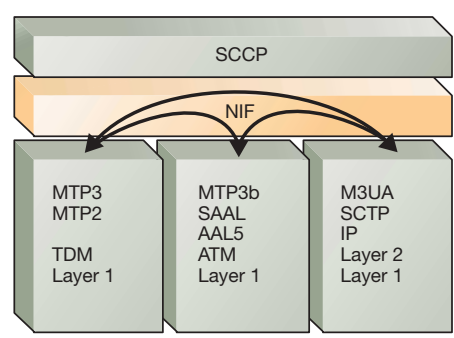


Figure 6 Built-in support for signaling gateway.

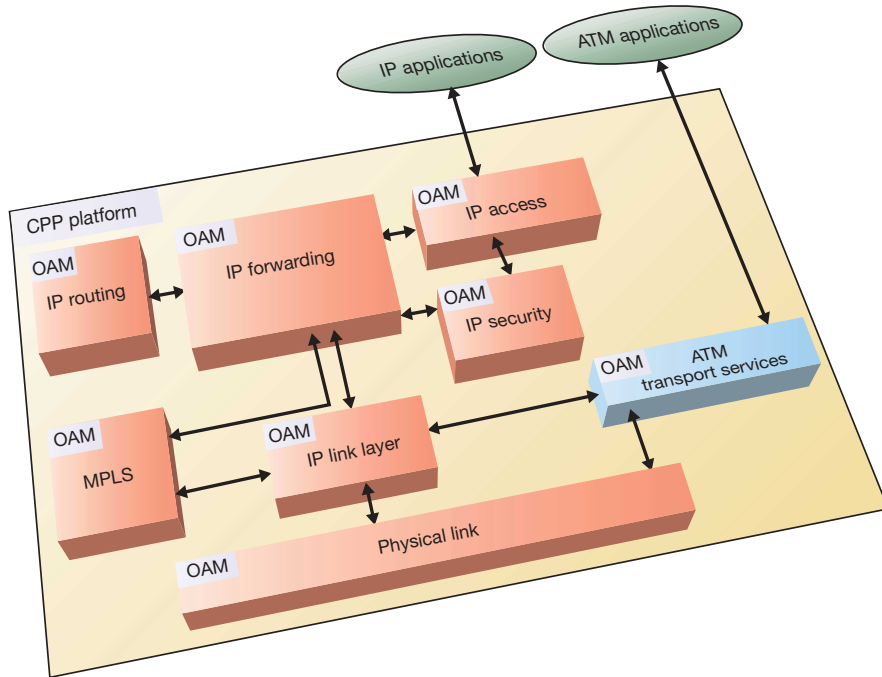


Figure 7  
The CPP IP architecture.

and IPv6 packets. Forwarding modules (Box C) implemented in software or in dedicated hardware circuits can be put on any board in the CPP system and are interconnected by the CPP space switch. Hardware-

### BOX C, FORWARDING MODULES

Forwarding modules implement the functionality needed to forward or terminate an IP packet. They also provide the functionality for resolving dependencies on connected routers (the data part) and hosts (the control part).

Figure 8 shows the data part of a wire-speed forwarding module. Most of the functionality is implemented in dedicated hardware or network processors. In addition, an exception handler executes on a standard processor. The exception handler provides the functionality needed for dealing with some infrequent packet types. If, during any step in the forwarding process, an exception is identified, the packet is handed over to the exception handler.

Packets enter the forwarding module from an incoming link. Each packet is classified according to the type of service it requires. This classification might involve metering actual load. The type-of-service field in the packet might be changed as a result of the classification. The classification also serves as a sort of firewall, and might result in having the packet dropped.

The packet is next analyzed to find out if it

should be terminated in the node. If so, a point of termination in the node is determined and the packet is put into one of several queues to the switch. Otherwise, the packet is further analyzed to determine

- the best next hop in the network; and
- various related parameters. One parameter contains the outgoing link and the point in the node where the forwarding module responsible for this link is located.

The packet is then put into one of the switch queues. After having passed the switch, the packet arrives at the forwarding module of the appointed outgoing link where it enters a second classification step, which might result in the packet being assigned a different class.

Finally, the packet enters the queuing system to the outgoing link. The queuing system has several queues served by a configurable, weighted, fair-queuing scheduler. The system also has advanced dropping mechanisms to handle overload situations on the link. The treatment a packet receives in the queuing system is determined by the class to which the packet has been assigned.

based forwarding modules can achieve wire-speed forwarding. The IP forwarding subsystem also includes packet classification and filtering, including DiffServ queuing.

To forward an IP packet, the forwarding modules need a forwarding table that is calculated in the IP routing subsystem using static (configured) routes and dynamic routing protocols, such as the open shortest path first (OSPF), routing information protocol (RIP), and border gateway protocol (BGP). The routing protocol handlers monitor the network topology, and the routing table manager calculates a forwarding table at start-up and when the network topology changes. Forwarding information from the IP routing subsystem (Box D) is communicated to all forwarding modules in the CPP node using the dedicated forwarding information base (FIB) interface.

Telecommunications applications that use CPP IP transport services need to terminate and originate large amounts of IP traffic that comes from or is sent to the IP network. The IP access subsystem enables applications to use IP hosts in CPP. A user can access multiple IP hosts on every processor board in the system: each host is identified by a unique IP address. The hosts can handle IPv4 and IPv6, user datagram protocol (UDP), transmission control protocol (TCP), and stream control transmission protocol (SCTP) termination. If necessary, the hosts can be made robust by means of the moveable host concept (see also section on Robustness).

An automatic configuration mechanism simplifies host configuration. The IP access subsystem uses the IP forwarding subsystem to send and receive packets. For low-speed applications, CPP provides a distributed host mechanism whereby one IP host with one IP address can be distributed and used on multiple processor boards.

IP transport is also used for highly confidential traffic, such as control signaling and operation and maintenance (O&M) traffic. To guarantee the integrity of this traffic, the IP security subsystem provides tunnel- and transport-mode encryption and decryption of IP packets. The IP security encryption/decryption engines can be distributed on multiple processors or dedicated hardware circuits to yield greater capacity.

The CPP architecture has been prepared for the introduction of MPLS—CPP can serve as a label edge router or label switch router. However, modifications will need to be made to the forwarding modules, and ad-

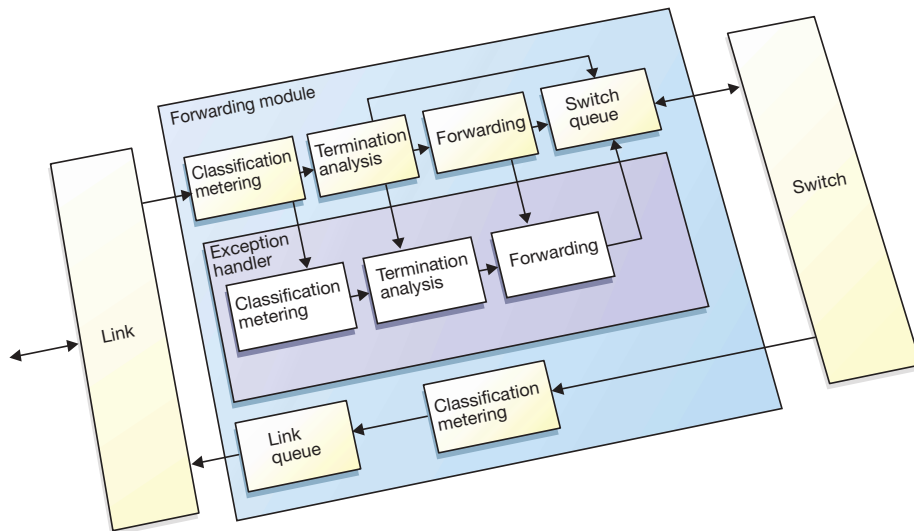


Figure 8  
Internal architecture of the forwarding module.

ditional signaling protocols will be introduced, including the

- resource reservation protocol – traffic engineering (RSVP-TE); and
- CR-LDP.

Likewise, interworking will be introduced between IP routing and MPLS.

Together with the physical layer subsystem, the IP link layer subsystem provides access to the external network. For example, the IP link layer subsystem supports 10/100 Mbit Ethernet, Gigabit Ethernet, point-to-point protocol (PPP), IP over ATM, and frame relay. Likewise, the physical layer supports STM-1/OC3, E1/J1/T1 and Ethernet. The IP link layer functionality is connected to the IP forwarding subsystem using the generic link interface.

## Scalability

CPP is uniquely scalable. It can be used in small applications (such as a small RBS) as well as large RNC and media gateway nodes. Indeed, virtually every aspect of scalability is covered by CPP: high-end and low-end nodes, payload capacity, processing capacity, number of routes, route updating capacity, number of physical links, link capacity, and cost.

In general, the IP functionality of CPP is composed of central functions and local functions. A central function solely exists in one instance for each virtual IP router, whereas a local function might exist in sev-

eral instances. With few exceptions, the scaling of local functions is quite straightforward. By contrast, the scaling of a central function is an intricate matter since central functions have a single, consistent information base. Distribution is not precluded but must be accomplished without incurring excessive costs in terms of memory and capacity.

### Payload handling capacity

CPP forwarding is based on interacting forwarding modules. Each forwarding module is a local function that handles a number of links and interfaces. Each processing entity (processor) and printed board assembly has its own forwarding and link module, which means that forwarding and link capacity are not adversely affected when new hardware is added. The only potential bottleneck is the intercommunication capacity of the forwarding module, which is based on the CPP space switch. At present, the capacity within the subrack is 16 Gbit/s, which is sufficient to support some 400,000 IP-borne voice calls. Moreover, multiple subracks can be interconnected to form very large nodes that constitute a single coherent system. As with forwarding, IP access mainly consists of local functions and scales smoothly when new hardware is added.

### Forwarding information base

The number of routes to be handled directly affects the size of the forwarding infor-

mation base (FIB). A very large FIB makes it unfeasible to house the entire forwarding table in fast path-forwarding logic. Therefore, CPP provides a caching scheme which forwards packets that fall outside the scope of the cache. At present, the fast path-forwarding path supports up to 64,000 routes.

### Routing protocols

The central functions that demand the most processing power in CPP are the routing protocol handlers (OSPF, BGP and so on). A shortage of processing power for routing results in unacceptably long convergence time—that is, the time it takes to regenerate a routing table when the topology has changed.

Certain measures affecting network layer configuration can significantly reduce the processing load on the individual routers. These measures are hierarchical network topology, appropriate address allocation, route summarization, and area subdivision. CPP supports these measures as well as scalability at the node level.

### Allocation of routing protocol handlers

Different routing protocol handlers (RPH) can be allocated to different processors. Every RPH is mastered by a single routing table manager. However, other scaling solutions are necessary when a single RPH requires more capacity than can be provided by one processor.

### Adding a virtual router

Where the network is concerned, the addition of a virtual router is no different from the addition of a physical router. In either case, the network becomes more complex and the RPHs must work harder to keep track of the network topology. The advantages in the local node are that the interfaces can be partitioned between virtual routers, and functions that belong to different virtual routers can be allocated to different processors. In addition, costs can be avoided provided that the virtual-router interconnecting link is implemented efficiently.

### Distributing a single RPH

A single RPH can be distributed to some extent by allocating certain sub-functions to different processors. The exact allocation of sub-functions varies according to the types of routing protocol in use. For OSPF, the shortest path calculation must be executed in one processor, whereas the handling of interfaces can be distributed.

### Low-end scalability

For CPP to meet the requirements of radio base stations it must support low-end scalability. As a consequence, the CPP IP architecture has been designed to accommodate IP functionality on a single PBA and to eliminate the switch. Another feature is the FIB caching mechanism mentioned above. These features keep the cost of fast forwarding-path logic to a minimum even in relatively complex networks with multiple routes.

## Robustness

The fully distributed IP-forwarding and IP-termination mechanisms in CPP are very robust. The failure of a board solely affects the forwarding and termination of IP packets on that board. All other boards continue forwarding and terminating as if nothing had happened, except that packets are not forwarded to the failed board. Should a network interface board or link fail, the routing protocols initiate link protection switching or automatic rerouting.

The robustness principles applied in CPP make it possible to define reliable programs that execute in the main processor cluster. These programs can be run on a standby processor should the main processor fail. The CPP IP functionality employs this fail-over concept for all central functions, such as the routing protocols, the central parts of IP ac-

## BOX D, IP ROUTING

The Internet is made up of several autonomous systems (AS), each of which is operated by an Internet service provider (ISP) or an organization (Figure 9). In telecommunications, each IP-based radio access network or core network can be seen as an autonomous system. Interior gateway routing protocols, such as RIP, OSPF or ISIS, are used inside autonomous systems to automatically create forwarding tables to be used by the routers.

OSPF is one of the most popular interior gateway protocols. This link state routing protocol discovers its neighboring routers and learns their network addresses by sending out 'Hello' packets on all its interfaces. The information it receives is stored in a topology database. OSPF then sends out a link state advertisement (LSA) packet indicating the status of its own links. The LSAs are 'flooded' by the other routers in order to reach every router in

the AS. Based on this input, OSPF calculates the shortest path to all routers and builds an optimal forwarding table. Whenever a change in network topology takes place—that is, if a link or a router goes down—new LSAs are flooded through the autonomous system. OSPF in each router updates the topology database and recalculates the forwarding table.

Some autonomous systems can be very large and difficult to manage. Likewise, the LSA flooding can generate a heavy load in a large autonomous system. To reduce load and simplify management, OSPF allows autonomous systems to be divided into OSPF areas. Routers connected to one area need only have detailed knowledge about the topology of that area. All areas in an autonomous system are interconnected by the backbone area. The routers connected to more than one area are called area border routers.

cess, IP security, IP forwarding, and all O&M functions. Should the processor on which the routing protocols execute fail, then the routing protocols are simply transferred to the standby processor. In the interim, while the protocols are being transferred and a new forwarding table is being built, the forwarding of packets on all other boards continues uninterrupted using the most recent forwarding table.

In coming releases of CPP, the standby routing protocol will operate in a listening mode and maintain an updated standby routing table.

The internal hosts in CPP, which applications use for terminating and originating IP traffic, can be distributed on any processor board in a CPP node. Each host is identified by a unique IP address. Should a board fail, CPP automatically transfers the host (and IP address) to another board. At the same time, the forwarding table is recalculated and traffic destined for the host is forwarded to the new location.

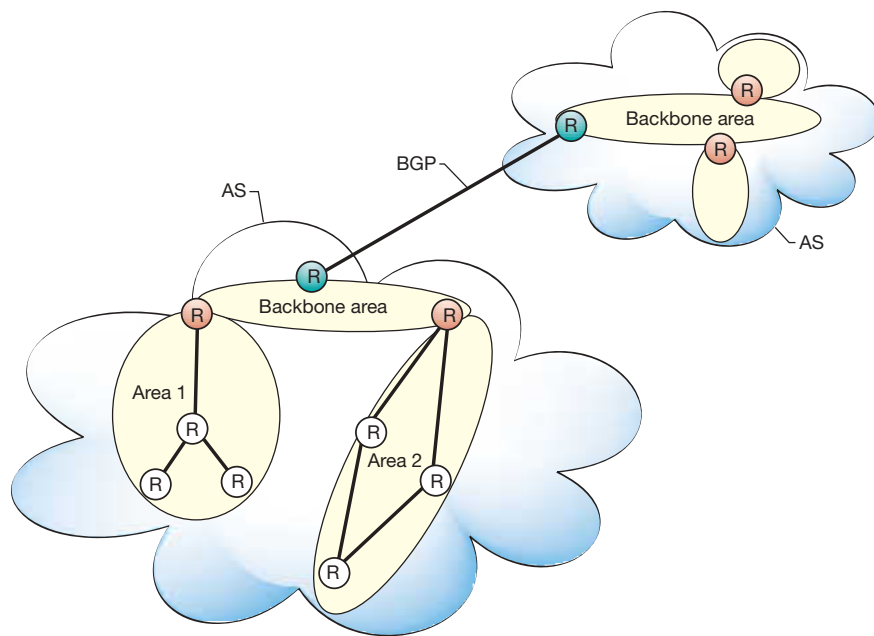
If an application requires it to do so, CPP can configure a robust Ethernet IP interface that uses a primary and secondary physical port. Should the primary port fail, the secondary port is activated using the IP address that the primary port had.

## Conclusion

Ericsson knows that operators want vendors to offer solutions and migration stories that are cost-effective and easy to maintain. Consequently, Ericsson offers telecommunications nodes with built-in transport capabilities for TDM, ATM and now, IP. Ericsson also knows that network operators want a compact and cost-effective solution that has a consistent network-management interface. Accordingly, Ericsson will make pure IP routers obsolete in operator networks.

The CPP IP architecture is composed of several subsystems that interwork with each other through well-defined interfaces:

- The IP forwarding subsystem provides fully distributed forwarding of IPv4 and IPv6 packets.
- The IP routing subsystem calculates a forwarding table, which forwarding modules use to forward IP packets.
- The IP access subsystem enables applications to use IP hosts in CPP.
- An automatic configuration mechanism simplifies host configuration.
- The IP security subsystem guarantees the integrity of sensitive traffic.



**Figure 9**  
Example of section of the Internet consisting of two autonomous systems (AS), where one AS is divided into three OSPF areas. The border gateway protocol (BGP) is used to convey routing information between the two autonomous systems.

The CPP architecture has been prepared for the introduction of MPLS—CPP can serve as a label edge router or label switch router. Interworking will be introduced between IP routing and MPLS. Together with the physical layer subsystem, the IP link layer subsystem provides access to the external network.

CPP is uniquely scalable. It can be used in small applications and large RNC and media gateway nodes. Virtually every aspect of scalability is covered by CPP: high-end and low-end nodes, pay-load capacity, processing capacity, number of routes, route updating capacity, number of physical links, link capacity, and cost.

The fully distributed IP-forwarding and IP-termination mechanisms in CPP are very robust. The robustness principles applied in CPP make it possible to define reliable programs that execute in the main processor cluster.